IEC 62734

Edition 1.0 2014-10

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Industrial networks – Wireless communication network and communication profiles – ISA 100.11a

Réseaux industriels – Réseau de communication sans fil et profils de communication – ISA 100.11a

IEC 62734:2014-10(en-fr)

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00
info@iec.ch
www.iec.ch

About the IEC
The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications
The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

IEC Catalogue - webstore.iec.ch/catalogue
The stand-alone application for consulting the entire bibliographical information on IEC International Standards, Technical Specifications, Technical Reports and other documents. Available for PC, Mac OS, Android Tablets and iPad.

IEC publications search - www.iec.ch/searchpub
The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished
Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and also once a month by email.

Electropedia - www.electropedia.org
The world's leading online dictionary of electronic and electrical terms containing more than 30 000 terms and definitions in English and French, with equivalent terms in 14 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC Glossary - std.iec.ch/glossary
More than 55 000 electrotechnical terminology entries in English and French extracted from the Terms and Definitions clause of IEC publications issued since 2002. Some entries have been collected from earlier publications of IEC TC 37, 77, 86 and CISPR.

IEC Customer Service Centre - webstore.iec.ch/csc
If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: csc@iec.ch.

A propos de l'IEC
La Commission Electrotechnique Internationale (IEC) est la première organisation mondiale qui élabore et publie des Normes internationales pour tout ce qui a trait à l'électricité, à l'électronique et aux technologies apparentées.

A propos des publications IEC
Le contenu technique des publications IEC est constamment revu. Veuillez vous assurer que vous possédez l'édition la plus récente, un corrigendum ou amendement peut avoir été publié.

Catalogue IEC - webstore.iec.ch/catalogue
Application autonome pour consulter tous les renseignements bibliographiques sur les Normes internationales, Spécifications techniques, Rapports techniques et autres documents de l'IEC. Disponible pour PC, Mac OS, tablettes Android et iPad.

Recherche de publications IEC - www.iec.ch/searchpub
La recherche avancée permet de trouver des publications IEC en utilisant différents critères (numéro de référence, texte, comité d'études,…). Elle donne aussi des informations sur les projets et les publications remplacées ou retirées.

IEC Just Published - webstore.iec.ch/justpublished
Restez informé sur les nouvelles publications IEC. Just Published détaille les nouvelles publications parues. Disponible en ligne et aussi une fois par mois par email.

Electropedia - www.electropedia.org
Le premier dictionnaire en ligne de termes électroniques et électriques. Il contient plus de 30 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans 14 langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International (IEV) en ligne.

Glossaire IEC - std.iec.ch/glossary
Plus de 55 000 entrées terminologiques électrotechniques, en anglais et en français, extraites des articles Termes et Définitions des publications IEC parues depuis 2002. Plus certaines entrées antérieures extraites des publications des CE 37, 77, 86 et CISPR de l'IEC.

Service Clients - webstore.iec.ch/csc
Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions contactez-nous: csc@iec.ch.

IEC 62734

Edition 1.0 2014-10

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Industrial networks – Wireless communication network and communication profiles – ISA 100.11a

Réseaux industriels – Réseau de communication sans fil et profils de communication – ISA 100.11a

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

PRICE CODE
CODE PRIX XH

ICS 25.040; 33.040; 35.100

ISBN 978-2-8322-1874-7

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

INDUSTRIAL NETWORKS – WIRELESS COMMUNICATION NETWORK AND COMMUNICATION PROFILES – ISA 100.11A

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.

International Standard IEC 62734 has been prepared by subcommittee 65C: Industrial networks, of IEC technical committee 65: Industrial-process measurement, control and automation.

This International Standard is based on ISA 100.11a:2011.

The reader's attention is drawn to the fact that Annex V lists all of the "in-some-country" clauses on differing practices of a less permanent nature relating to the subject of this standard.

This first edition cancels and replaces the IEC/PAS 62734 published in 2012. This edition constitutes a technical revision.

The text of this standard is based on the following documents:

FDIS	Report on voting
65C/778/FDIS	65C/788/RVD

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

The committee has decided that the contents of this publication will remain unchanged until the stability date indicated on the IEC web site under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

0 Introduction

0.1 General

This standard provides specifications in accordance with the OSI Basic Reference Model, ISO/IEC 7498–1, (e.g., PhL, DL, etc.), and also provides security and management (including network and device configuration) specifications for wireless devices serving Annex C's usage classes 1 through 5, and potentially class 0, for fixed, portable, and moving devices.

This standard is intended to provide reliable and secure wireless operation for non-critical monitoring, alerting, supervisory control, open loop control, and closed loop control applications. This standard defines a protocol suite, including system management, gateway considerations, and security specifications, for low-data-rate wireless connectivity with fixed, portable, and slowly-moving devices, often operating under severe energy and power constraints. The application focus is the performance needs of process automation monitoring and control where end-to-end communication latencies on the order of at least 100 ms can be tolerated.

To meet the needs of industrial wireless users and operators, the technology specified in this document provides robustness in the presence of interference found in harsh industrial environments or caused by wireless systems not covered by this international standard. As described in Clause 4, this standard addresses coexistence with other wireless devices anticipated in the industrial workspace, such as cell phones and devices based on IEC 62591 (based on WirelessHART™[1]), IEC 62601 (based on WIA-PA), IEEE 802.11 (WiFi), IEEE 802.15, IEEE 802.16 (WiMax), and other relevant standards. Furthermore, this standard supports interoperability of devices compliant with this international standard, as described in Clause 5, in those aspects of operation that are covered by this international standard.

This standard does not define or specify plant infrastructure or its security or performance characteristics. However, it is important that the security of the plant infrastructure be assured by the end user.

0.2 Document structure

This document is organized into clauses focused on unique network functions and protocol suite layers. The clauses describe system, system management, security management, physical layer, data-link layer, network layer, transport layer, application layer, and provisioning. Generic considerations that apply to protocol gateways are also included, though specifications of specific protocol gateways are not. Each clause describes a functionality or protocol layer and dictates the behavior required for proper operation. When a clause describes behaviors related to another function or layer, a reference to the appropriate other clause is supplied for further information.

The mandatory and optional communication protocols defined by this document are referred to as native protocols, while those protocols used by other networks such as legacy fieldbus communication protocols are referred to as foreign protocols.

0.3 Potentially relevant patents

The International Electrotechnical Commission (IEC) draws attention to the fact that it is claimed that compliance with this document may involve the use of multiple patents:

a) concerning elliptic curve (asymmetric) cryptography, given in 7.4.6 and 7.2.2.3;

---

[1] Property of the HART Communication Foundation. This information is given for the convenience of users of the standard and does not constitute an endorsement of the trademark holder or any related products. Compliance to this profile does not require use of the registered trademark. Use of the trademarks requires permission of the trade name holder.

b) concerning synchronizing clocks and assessing link quality, given in 9.1.9.3 and 9.1.15;

c) concerning unspecified subject areas;

d) concerning wireless provisioning, and selection and routing among multiple gateways.

IEC takes no position concerning the evidence, validity and scope of these patent rights.

The holders of these patent rights have assured the IEC that they are willing to negotiate licences either free of charge (free) or under reasonable and non-discriminatory terms and conditions (RAND) with applicants throughout the world. In this respect, the statements of the following holders of those patent rights are registered with IEC.

Information on these patent rights and their licensing may be obtained from:

a)	Certicom Corporation 4701 Tahoe Blvd, Bldg A L4W 0B5 Mississauga, ON CANADA Attn: Patent licensing Licensing terms: presumably RAND Relevant patents: unknown; not stated by patent holder	b)	NIVIS LLC 1000 Circle 75 Pkwy, Suite 300 Atlanta, GA 30339-6051 USA Attn: Patent licensing Licensing terms: RAND Relevant patents: – US 20100027437 – US 20100098204
c)	General Electric 1 Research Cir Schenectady, NY 12309-1027 USA Attn: Patent licensing Licensing terms: presumably RAND, reciprocity Relevant patents: unknown; not stated by patent holder	d)	Yokogawa Electric Corporation 2-9-32 Nakachou, Musashina-shi Tokyo JAPAN Attn: Patent licensing Licensing terms: RAND, reciprocity Relevant patents: – JP 4129749 – US 8005514 – US 8031727 – US 8305927 – US 2009080394
The above patent holders, patents, and licensing terms are those declared to the IEC as relevant to IEC 62734, as of the date of preparation of this text.			

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified above. IEC shall not be held responsible for identifying any or all such patent rights.

ISO (http://www.iso.org/patents) and IEC (http://patents.iec.ch) maintain on-line databases of patents relevant to their standards. Users are encouraged to consult these databases for the most up-to-date information concerning patents.

INDUSTRIAL NETWORKS – WIRELESS COMMUNICATION NETWORK AND COMMUNICATION PROFILES – ISA 100.11A

1 Scope

This International Standard specifies a method of reliable and secure wireless operation for non-critical monitoring, alerting, supervisory control, open loop control, and closed loop control applications. This standard defines a protocol suite, including system management, gateway considerations, and security specifications, for low-data-rate wireless connectivity with fixed, portable, and slowly-moving devices, often operating under severe energy and power constraints. The application focus of this standard is the performance needs of process automation monitoring and control, where end-to-end communication delays on the order of 100 ms can be tolerated.

This standard specifies the following:

- physical layer service definition and protocol specification;
- data-link layer service definition and protocol specification;
- network layer service definition and protocol specification;
- transport layer service definition and protocol specification;
- application layer service definition and protocol specification, including support for protocol tunneling and gateways;
- security and security management;
- provisioning and configuration;
- network management; and
- additive communication role profiles (i.e., one or more can be selected concurrently).

Functionality above the application layer of the OSI Basic Reference Model, such as the so-called User Layer and different profiles for functionality at that layer, is not addressed. However, it is discussed briefly in Annex A.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

NOTE See the Bibliography for non-normative references.

ISO/IEC 646, *Information technology – ISO 7-bit coded character set for information interchange*

ISO/IEC 10731, *Information technology – Open Systems Interconnection – Basic Reference Model – Conventions for the definition of OSI services*

ISO/IEC 18033-3, *Information technology – Security techniques – Encryption algorithms – Part 3: Block ciphers*

ISO/IEC 19772, *Information technology – Security techniques – Authenticated encryption*

ANSI X9.63:2011, *Public Key Cryptography for the Financial Services Industry – Key Agreement and Key Transport Using Elliptic Curve Cryptography*

IETF RFC 2460:1998, *Internet Protocol, Version 6 (IPv6) Specification*

IETF RFC 2464, *Transmission of IPv6 Packets over Ethernet Networks*

IETF RFC 2529, *Transmission of IPv6 over IPv4 Domains without Explicit Tunnels*

IETF RFC 3168, *The Addition of Explicit Congestion Notification (ECN) to IP*

IETF RFC 4213, *Basic Transition Mechanisms for IPv6 Hosts and Routers*

IETF RFC 4291:2006, *IP Version 6 Addressing Architecture*

IETF RFC 4944, *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*

IETF RFC 6282:2011, *Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks*

IETF RFC 6298, *Computing TCP's Retransmission Timer*

IEEE 802.15.4™:2011[2], *IEEE Standard for Information technology — Telecommunications and information exchange between systems — Local and metropolitan area networks — Specific requirements – Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)*

SEC 1:2009, *Elliptic Curve Cryptography, version 2*, available at http://www.secg.org

SEC 4, *Elliptic Curve Qu-Vanstone Implicit Certificate Scheme (ECQV), version 0.97*, available at http://www.secg.org

3 Terms, definitions, abbreviated terms, acronyms, and conventions

For the purposes of this document, the following terms, definitions, abbreviations, acronyms and conventions apply.

3.1 Terms and definitions

3.1.1 (N)-layer and other terms and definitions from the open systems interconnection Basic Reference Model

3.1.1.1
abstract syntax
specification of (N)-PDUs by using notation rules which are independent of the encoding technique used to represent them

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 7.1.1.2, modified – generalized to any layer]

---

2 Property of IEEE, http://www.ieee.org.

3.1.1.2
accountability
property that ensures that the actions of an entity may be traced uniquely to the entity

[SOURCE: ISO 7498-2:1989, 3.3.3]

3.1.1.3
acknowledgment
function of the (N)-layer which allows a receiving (N)-entity to inform a sending (N)-entity of the receipt of an (N)-PDU

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.8.1.16]

3.1.1.4
application-entity
active element, within an application process, embodying a set of capabilities that is pertinent to OSI and that is defined for the AL, that corresponds to a specific application-entity-type (without any extra capabilities being used)

Note 1 to entry: This is a slight specialization of (N)-entity, because the AL includes non-OSI-relevant application functions. Each application-entity represents one and only one process in the open system interconnection environment.

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 7.1.1.1]

3.1.1.5
application-management
functions in the AL related to management of OSI application-processes

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 8.1.1]

3.1.1.6
association
cooperative relationship between system entities, usually for the purpose of transferring information between them

[SOURCE: IEC TS 62443-1-1:2009, 3.2.7]

3.1.1.7
(N)-association
cooperative relationship among (N)-entity-invocations

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.3.1.1]

3.1.1.8
authorization
granting of rights, which includes the granting of access based on access rights

[SOURCE: ISO 7498-2:1989, 3.3.10]

3.1.1.9
availability
property of being accessible and useable upon demand by an authorized entity

[SOURCE: ISO 7498-2:1989, 3.3.11]

3.1.1.10
blocking
function performed by an (N)-entity to map multiple (N)-SDUs into one (N)-PDU

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.8.1.11]

3.1.1.11
centralized-multi-endpoint-connection
multi-endpoint-connection where data sent by the entity associated with the central-connection-endpoint is received by all other entities, while data sent by the other entities is received by only the central entity

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.8.1.2]

3.1.1.12
ciphertext
data produced through the use of encipherment so that the semantic content of the resulting data is not available

Note 1 to entry: See cleartext, plaintext.

Note 2 to entry: Relative to a PDU, ciphertext is information in a PDU that is subject to obscuration by encryption, in its post-encryption pre-decryption obscured form.

[SOURCE: ISO 7498-2:1989, 3.3.1]

3.1.1.13
cleartext
<generic> intelligible data, the semantic content of which is available

[SOURCE: ISO 7498-2:1989, 3.3.15]

3.1.1.14
cleartext
<communications-protocol-specific> information in a PDU that is not subject to obscuration by encryption

Note 1 to entry: Relative to a PDU, cleartext is information in the PDU that is not subject to obscuration by encryption, that when present in the PDU is always present in its unobscured form.

3.1.1.15
compromise
violation of computer security whereby programs or data may have been modified, destroyed, or made available to unauthorized entities

[SOURCE: ISO/IEC 2382-8:1998, 08.05.11]

3.1.1.16
concatenation
function performed by an (N)-entity to map multiple (N)-PDUs into one (N-1)-SDU

Note 1 to entry: Blocking and concatenation, though similar (they both permit grouping of data-units) serve different purposes. For instance, concatenation permits the (N)-layer to group one or several acknowledgment (N)-PDUs with one (or several) (N)-PDUs containing user-data. This would not be possible with the blocking function only. Note also that the two functions are combinable so that the (N)-layer performs blocking and concatenation.

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.8.1.13]

3.1.1.17
concrete syntax
those aspects of the rules used in the specification of data which embody a specific representation of that data

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 7.2.1.1]

3.1.1.18
confidentiality
property that information is not made available or disclosed to unauthorized individuals, entities, or processes

Note 1 to entry: In a general information security context, confidentiality preserves authorized restrictions on information access and disclosure, including means for preserving personal privacy and proprietary information.

[SOURCE: ISO 7498-2:1989, 3.3.16]

3.1.1.19
(N)-connection
association requested by an (N+1)-layer entity for the transfer of data between two or more (N+1)-entities

Note 1 to entry: The association is established by the (N)-layer and provides explicit identification of a set of (N)-data-transmissions and agreement concerning the (N)-data-transmission services to be provided for the set.

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.3.1.2]

3.1.1.20
(N)-connection endpoint
terminator at one end of an (N)-connection within an (N)-service-access-point

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.3.1.3]

3.1.1.21
(N)-connection-endpoint-identifier
identifier of an (N)-connection-endpoint which can be used to identify the corresponding (N)-connection at an (N)-SAP

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.4.1.5]

3.1.1.22
(N)-connection-endpoint-suffix
that part of an (N)-connection-endpoint-identifier which is unique within the scope of an (N)-SAP

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.4.1.6]

3.1.1.23
(N)-connection-mode-transmission
(N)-data-transmission in the context of an (N)-connection

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.3.1.17]

3.1.1.24
(N)-connectionless-mode-transmission
(N)-data-transmission not in the context of an (N)-connection and not required to maintain any logical relationship between (N)-SDUs

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.3.1.18]

3.1.1.25
correspondent-(N)-entities
(N)-entities with an (N-1)-connection between them

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.3.1.5]

3.1.1.26
cryptanalysis
analysis of a cryptographic system and/or its inputs and outputs to derive confidential variables and/or sensitive data including cleartext

[SOURCE: ISO 7498-2:1989, 3.3.18]

3.1.1.27
data integrity
property that data has not been altered or destroyed in an unauthorized manner

[SOURCE: ISO 7498-2:1989, 3.3.21]

3.1.1.28
data-origin authentication
corroboration that the source of data received is as claimed

[SOURCE: ISO 7489-2:1989, 3.3.22]

3.1.1.29
(N)-data transmission
(N)-facility that conveys SDUs from one (N+1) layer entity to one or more (N+1) entities

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.3.1.9]

3.1.1.30
deblocking
function performed by an (N)-entity to identify multiple (N)-SDUs which are contained in one (N)-PDU

Note 1 to entry: In the absence of error, deblocking is the reverse function of blocking.

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.8.1.12]

3.1.1.31
decentralized-multi-endpoint-connection
multi-endpoint-connection where data sent by an entity associated with a connection-endpoint is received by all other entities

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.8.1.3]

3.1.1.32
decryption
reversal of a corresponding reversible encipherment

[SOURCE: ISO 7498-2:1989, 3.3.23]

3.1.1.33
demultiplexing
function performed by an (N)-entity which identifies (N)-PDUs for more than one (N)-connection within an (N-1)-connection

Note 1 to entry: In the absence of error, demultiplexing is the reverse function of multiplexing.

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.8.1.5]

3.1.1.34
digital signature
data appended to, or a crypto graphic transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery e.g. by the recipient

[SOURCE: ISO 7498-2:1989, 3.3.26]

3.1.1.35
(N)-entity
active element within an (N)-subsystem embodying a set of capabilities defined for the (N)-layer that corresponds to a specific (N)-entity-type (without any extra capabilities being used)

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.2.1.11]

3.1.1.36
(N)-entity-invocation
specific utilization of part or all or all of the capabilities of a given (N)-entity (without any extra capabilities being used)

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.2.1.12]

3.1.1.37
(N)-entity-type
description of a class of (N)-entities in terms of a set of capabilities defined for the (N)-layer

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.2.1.10]

3.1.1.38
(N)-interface-control-information
information transferred locally between an (N+1)-entity and an (N)-entity to coordinate their joint operation

3.1.1.39
(N)-layer
subdivision of the OSI architecture, constituted by subsystems of the same rank (N)

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996,5.2.1.2]

3.1.1.40
(N)-layer-management
functions related to the management of the (N)-layer partly performed in the (N)-layer itself according to the (N)-protocol of the layer (activities such as activation and error control) and partly performed as a subset of systems-management

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 8.1.6]

3.1.1.41
multi-endpoint-connection
connection with more than two connection-endpoints

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.3.1.4]

3.1.1.42
multiplexing
function performed by an (N)-entity in which one (N-1)-connection is used to support more than one (N)-connection

Note 1 to entry: The term multiplexing is also used in a more restricted sense to the function performed by the sending (N)-entity while the term demultiplexing is used to the function performed by the receiving (N)-entity.

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.8.1.4]

3.1.1.43
password
confidential authentication information, usually composed of a string of characters

[SOURCE: ISO 7498-2:1989, 3.3.39]

3.1.1.44
peer-(N)-entities
entities within the same (N)-layer

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.2.1.3]

3.1.1.45
peer-entity authentication
corroboration that a peer entity in an association is the one claimed

[SOURCE: ISO 7489-2:1989, 3.3.40]

3.1.1.46
(N)-protocol
set of rules and formats (semantic and syntactic) that determines the communication behavior of (N)-entities in the performance of (N)-functions

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.2.1.9]

3.1.1.47
(N)-protocol-addressing-information
those elements of (N)-PCI which contain addressing information

[SOURCE: ISO/IEC 7498-3:1997, 3.4.20]

3.1.1.48
(N)-protocol-control-information
information exchanged between (N)-entities to coordinate their joint operation

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.6.1.1]

3.1.1.49
(N)-protocol-data-unit
unit of data specified in an (N)-protocol and consisting of (N)-protocol-control-information and possibly (N)-user-data

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.6.1.3]

3.1.1.50
(N)-protocol-version-identifier
identifier conveyed between correspondent (N)-entities which allows the selection of the version of an (N)-protocol

Note 1 to entry: The definition of a new (N)-protocol-version-identifier presupposes a minimal common knowledge of the (N)-protocol identified by the preceding (N)-protocol-version-identifier. When such a minimal common knowledge cannot be achieved, the (N)-protocols are considered to be independent and different.

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.8.1.18]

3.1.1.51
quality of service
<generic> collective effect of service performance which determines the degree of satisfaction of a user of the service

Note 1 to entry: The quality of service is characterized by the combined aspects of service support performance, service operability performance, serveability performance, service integrity and other factors specific to each service.

Note 2 to entry: ISO defines quality as the ability of a product or service to satisfy users' needs.

[SOURCE: IEC 61907:2009, 3.1.15]

3.1.1.52
quality of service
<data link service> negotiated parameters for a link, including

- priority;
- time windows for control messaging;
- acceptability of out-of-order message delivery; and
- acceptability of message delivery in partial increments

3.1.1.53
reassembling
function performed by an (N)-entity to map multiple (N)-PDUs into one (N)-SDU

Note 1 to entry: In the absence of error, reassembling is the reverse function of segmenting.

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.8.1.10]

3.1.1.54
recombining
function performed by an (N)-entity which identifies (N)-PDUs for a single (N)-connection in (N-1)-SDUs received on more than one (N-1)-connection

Note 1 to entry: In the absence of error, recombining is the reverse function of splitting.

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.8.1.7]

3.1.1.55
(N)-relay
(N)-function by means of which an (N)-entity forwards data received from one peer (N)-entity to another peer (N)-entity

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.3.1.6]

3.1.1.56
reset
function that sets the corresponding (N)-entities to a predefined state with a possible loss or duplication of data

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.8.1.17]

3.1.1.57
security label
marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource

[SOURCE: ISO 7498-2:1989, 3.3.49]

3.1.1.58
segmenting
function performed by an (N)-entity to map one (N)-SDU into multiple (N)-PDUs

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.8.1.9]

3.1.1.59
(N)-selector
that part of an (N)-address that is specific to the addressed (N)-subsystem, i.e., which identifies one or more (N)-SAPs within an end open system once that end open system is unambiguously identified

Note 1 to entry: Since the end open system is implicitly known at the Network layer, (N)-selectors are used above the Network layer, along with local information, to address the desired (N+1)-entity within the open system. (N)-selector values are exchanged between open systems as part of the (N)-PAI.

[SOURCE: ISO/IEC 7498-3:1997, 6.2.3]

3.1.1.60
separation
function performed by an (N)-entity to identify multiple (N)-PDUs which are contained in one (N-1)-SDU

Note 1 to entry: In the absence of error, separation is the reverse function of concatenation.

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.8.1.14]

3.1.1.61
sequencing
function performed by the (N)-layer to preserve the order of (N)-SDUs that were submitted to the (N)-layer

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.8.1.15]

3.1.1.62
(N)-service
capability of the (N)-layer and the layers beneath it, which is provided to (N+1) entities at the boundary between the (N)-layer and the (N+1) layer

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.2.1.5]

3.1.1.63
(N)-service access point
point at which (N)-services are provided by an (N)-entity to an (N+1)-entity

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.2.1.8]

3.1.1.64
(N)-service-access-point-address
(N)-address that is used to identify a single (N)-SAP

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.4.1.2]

3.1.1.65
(N)-service-data-unit
amount of information whose identity is preserved when transferred between peer (N+1)-entities and which is not interpreted by the supporting (N)-entities

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.6.1.4]

3.1.1.66
splitting
function within the (N)-layer by which more than one (N-1)-connection is used to support one (N)-connection

Note 1 to entry: The term splitting is also used in a more restricted sense to see the function performed by the sending (N)-entity while the term recombining is used to see the function performed by the receiving (N)-entity.

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.8.1.6]

3.1.1.67
system management
functions in the AL related to management of various OSI resources and their status across all layers of the OSI architecture

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 8.1.4]

3.1.1.68
transfer syntax
abstract and concrete syntax used in the transfer of data between open systems

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 7.2.1.2]

3.1.1.69
user application process
active process within the highest portion of the AL that is the user of OSI services

Note 1 to entry: The aspects of a UAP that need to be taken into account for the purpose of OSI are represented by one or more application-entities, of one or more application-entity-types, defined in ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 7.1.2.2 and 7.1.2.3.

Note 2 to entry: The collection of UAPs is sometimes referred to as the user layer, even though ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 7.1.2.1 states that the AL has no boundary with a higher layer. In the OSI Basic Reference Model, the AL includes the UAPs.

3.1.1.70
(N)-user-data
data transferred between (N)-entities on behalf of the (N+1)-entities for which the (N)-entities are providing services

[SOURCE: ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, 5.6.1.2]

3.1.2 Other terms and definitions

NOTE Sources of definitions that are otherwise unreferenced by this standard can be found in the Bibliography.

3.1.2.1
access control
means to ensure that access to assets is authorized and restricted based on business and security requirements

[SOURCE: ISO/IEC 27000:2014, 2.1]

3.1.2.2
adapter
specialization of a device that internally converts the protocol in the attached legacy device(s) to that of a network device compliant with this standard

3.1.2.3
alarm
condition that maintains a state until the condition clears, reported on change of state

EXAMPLE The occurrence of an alarm or of a return-to-normal condition that is of potential significance to a correspondent UAP.

3.1.2.4
alert
action of reporting an event condition or an alarm condition

3.1.2.5
end application
application
system or problem to which a computer is applied

3.1.2.6
application program
application
program that provides functionality to end users

3.1.2.7
application layer
application
highest protocol layer in the ISO/IEC Basic Reference Model in which types of applications are classified according to criticality

3.1.2.8
application process
element that performs the information processing for a particular application

3.1.2.9
asymmetric-key algorithm
asymmetric-key cryptographic algorithm
public key cryptographic algorithm
<information security> algorithm for performing encipherment or the corresponding decipherment in which the keys used for encipherment and decipherment differ

[SOURCE: ISO/IEC 10181-1:1996, 3.3.1]

3.1.2.10
authentication
<information security> verifying the identity of a user, process, or device, often as a prerequisite to allowing access to resources in an information system

Note 1 to entry: See data-origin authentication (3.1.1.28) and/or peer-entity authentication (3.1.1.45).

3.1.2.11
authentication code
<information security> full or truncated cryptographic checksum based on an appropriate security function

Note 1 to entry: See 3.1.2.97.

Note 2 to entry: This is also known as a message authentication code (MAC). It is called a message integrity code (MIC) when used in contexts where the acronym MAC has an alternate definition, such as in local area network standards.

[SOURCE: ISO/IEC 19790:2012, 3.8, modified by the “full or truncated” prefix]

3.1.2.12
backbone network
backbone subnet
network not specified by this standard, generally using IPv6 or IPv4 network technology, that is used for routing between the wireless industrial sensor (and actuator) network (WISN) of this standard and

a) connected back-end devices that are specified in part by this standard, such as system managers, security managers, and protocol gateways;
b) devices that natively support the wireless TL, AL and management protocols of this standard; and
c) other backbone routers on the same backbone subnet

3.1.2.13
backbone router
router that forwards between the wireless network of this standard and a higher-speed backbone network

3.1.2.14
backup
procedure, technique, or hardware used to help recover lost or destroyed data or to keep a system operating

[SOURCE: ISO 2382-12:1988, 12.01.17]

3.1.2.15
bandwidth
<analog domain> numerical difference between the upper and lower frequencies of a band of frequencies

Note 1 to entry: Analog bandwidth is expressed in Hz.

3.1.2.16
bandwidth
<digital domain> amount of data that can be passed along a communications channel within a given period of time

Note 1 to entry: Digital bandwidth is expressed in bit/s.

3.1.2.17
black channel
communication channel of a safety system that provides no safety functionality in addition to its basic communication capability

3.1.2.18
blacklist
list of RF channels upon which transmission is prohibited

Note 1 to entry: A blacklist is temporary or permanent, local or network-wide.

3.1.2.19
block cipher
<information security> cryptographic primitive that uses a symmetric key to create a key-dependent pseudorandom permutation of a fixed-size bit string

3.1.2.20
broadcast
transmission intended for all nodes

Note 1 to entry: Broadcast reception often is limited to specific layers, e.g., MAC or network layer.

Note 2 to entry: Many lower layer protocols do not provide an acknowledgment for broadcasts.

3.1.2.21
canonical transfer syntax
full transfer syntax
full encoding of an object for transfer between devices, before any compression

3.1.2.22
cipher
<information security> cryptographic technique used to protect the confidentiality of data, consisting of three component processes: an encryption algorithm, a decryption algorithm, and a method for generating keys

[SOURCE: attributed in other ISO/IEC standards to an unknown edition of ISO/IEC 18033-1, modified – slightly edited for readability.]

3.1.2.23
coexistence
ability of multiple systems to perform their tasks in a given environment where they may or may not be using a similar set of rules

3.1.2.24
compressed transfer syntax
encoding of an object for transfer between devices, after any compression

3.1.2.25
communications latency
delay between an initial communications event, such as making a transmit request, and a related subsequent communications event, such as message delivery at an intended recipient

3.1.2.26
communications reliability
degree tp which messages reach their final destination complete and uncorrupted and within acceptable latency

3.1.2.27
communications throughput
capacity of a communications channel, usually measured in message/s or octet/s

3.1.2.28
construction option
set of features that a device designer may choose to include in, or exclude from, a device

3.1.2.29
contract
agreement between the system manager and a device in the network involving the allocation of network resources by the system manager to support a particular communication need of that device

3.1.2.30
cryptographic algorithm
<information security> algorithm based upon the science of cryptography, including encryption algorithms, cryptographic hash algorithms, digital signature algorithms, and key agreement algorithms

Note 1 to entry: Examples of cryptographic algorithms are block and stream ciphers and keyed hashes. An unkeyed hash is not formally a cryptographic algorithm, although it often is a one-way function that has similar resistance to attack, and often is constructed from a cryptographic algorithm with a fixed key. The SHA family of hashes is so constructed.

[SOURCE: IEC TS 62443-1-1:2009, 3.2.34, modified – a note has been added.]

3.1.2.31
cryptographic key
key
<information security> mathematical value that is used

a) in an algorithm to generate ciphertext from plaintext or vice versa, and
b) to determine the operation of a cryptographic function (e.g., the synchronized generation of keying material), or a digital signature computation or validation

[SOURCE: IEC TS 62351-2:2008, 2.2.64]

3.1.2.32
cryptographic key component
key component
<information security> parameter(s) used in a security function to perform a cryptographic function

[SOURCE: ISO/IEC 19790:2012, 3.24]

3.1.2.33
cryptographic module
<information security> set of hardware, software, and/or firmware that implements appropriate security functions and is contained within the cryptographic boundary

[SOURCE: ISO/IEC 19790:2012, 3.25]

3.1.2.34
cryptoperiod
<information security> time span during which a specific key is authorized for use or in which the keys for a given system or application may remain in effect

[SOURCE: ISO/IEC 11568-4:2007, 3.9]

3.1.2.35
data authenticity
<information security> assurance about the source of information

[SOURCE: IEEE 802-15-4:2011, 3.1]

3.1.2.36
data key
data authenticating key
data encrypting key
<information security> cryptographic key used for the encipherment, decipherment or authentication of data

[SOURCE: ISO/IEC 11568-2:2012, 3.5]

3.1.2.37
deployment option
set of features that a device designer includes in a device, but which the end-user or their agent (e.g., a network security manager) can elect to employ or not employ

3.1.2.38
derived key
<information security> symmetric key that is derived from a prior symmetric key

Note 1 to entry: Such keys are usable to limit the cryptoperiod of any single key while meeting key archive requirements, provided that the independent key from which the derived key was derived (perhaps through many generations of derivation) has previously met those archive requirements.

3.1.2.39
key destruction
destruction
<information security> zeroisation or physical destruction of keying material so that it cannot be recovered

3.1.2.40
deterministic random bit generator
<information security> process used to generate an unpredictable series of bits that are random in the sense that there is no way to describe the generator's output that is more efficient than simply listing each entire output string

Note 1 to entry: Deterministic random bit generators have provable properties. The unpredictability of their output depends on the unpredictability of their initial seed and, for hybrid generators, the rate at which new unpredictable (high entropy) input is included relative to the number of output bits generated. See note to entry 1 of 3.1.2.102, non-deterministic random bit generator, for common sources of such unpredictability.

3.1.2.41
device security management object
application software within a device that acts as a local peer of a security manager

3.1.2.42
duocast
variant of unicast, wherein a second receiver is scheduled to overhear the DPDU and provides a second acknowledgment within a single D-transaction

Note 1 to entry: Duocast is shown graphically in Figure 84.

3.1.2.43
encrypted key
<information security> cryptographic key that has been encrypted using an approved security function with a key encryption key

Note 1 to entry: This process is used in order to disguise the value of the underlying plaintext key.

[SOURCE: ISO/IEC 19790:2012, 3.36]

3.1.2.44
encryption
<information security> reversible operation by a cryptographic algorithm converting data into ciphertext so as to hide the information content of the data

[SOURCE: ISO/IEC 9798-1:2010, 3.13]

3.1.2.45
entity
individual (person), organization, device or process

3.1.2.46
ephemeral key
<information security> cryptographic key that is generated for each execution of a key establishment process and that meets other requirements of the key type (e.g., unique to each message exchange or session)

Note 1 to entry: In some cases ephemeral keys are used more than once, within a single session (e.g., broadcast applications) where the originator generates only one ephemeral key pair per message and the private key (of that pair) is combined separately with each recipient's public key.

3.1.2.47
event
transient (i.e., stateless) condition, used to report when something happened

EXAMPLE The occurrence of an alarm or a return-to-normal condition that is of potential significance to a correspondent UAP.

3.1.2.48
field device
physical device designed to meet the rigors of plant operation that communicates via DPDUs and higher-layer protocols conforming to this standard

Note 1 to entry: These include routing devices, sensors, and actuators.

3.1.2.49
field network
configuration of two or more field devices interconnected by the wireless protocol defined by this standard

3.1.2.50
field router
router that is also a field device (i.e., not a backbone router), existing within a field network

3.1.2.51
foreign protocol application communication
optimized conveyance of PDUs or portions of PDUs from a first protocol within a second protocol by selective usage of caching, compression, address translation and proxy techniques

3.1.2.52
fragment, verb
segment
to break or separate into contiguous disjoint parts

Note 1 to entry: Fragment and fragmentation are the terms used by the IETF in internet protocol specifications to describe the OSI concepts of segment and segmentation. In this standard the terms fragment and segment are essentially synonymous, with fragment usually used at the network layer and sometimes at other protocol layers, while segment is usually used at the application layer and sometimes at other protocol layers.

3.1.2.53
fragment, noun
segment
one of the (N)-protocol-data-units resulting from the operation of segmenting

Note 1 to entry: Fragment and fragmentation are the terms used by the IETF in internet protocol specifications to describe the OSI concepts of segment and segmentation. In this standard the terms fragment and segment are essentially synonymous, with fragment usually used at the network layer and sometimes at other protocol layers, while segment is usually used at the application layer and sometimes at other protocol layers.

3.1.2.54 gateway
role (of a device) that acts as a protocol translator between an AE conforming to this standard and other, different AEs

3.1.2.55
hash-based message authentication code
<information security> message authentication code that uses an appropriate keyed-hash function

Note 1 to entry: This definition is generalized from the second construction of ISO/IEC 9797-2, which refers to the HMAC construction that is also specified in [US] FIPS 198.

[SOURCE: ISO/IEC 9797-2:2011, modified – generalized.]

3.1.2.56
hash function
<information security> function which maps strings of bits to fixed-size strings of bits, satisfying the following two properties: for a given output, it is computationally infeasible to find an input which maps to this output; for a given input, it is computationally infeasible to find a second input which maps to the same output

Note 1 to entry: Computational feasibility depends on the specific security requirements and environment.

Note 2 to entry: See the note 1 to entry of 3.1.2.30.

[SOURCE: ISO/IEC 9796-2:2010, 3.6, modified – notes to entry have been added.]

3.1.2.57
hash value
<information security> full or truncated result of applying a hash function to information

3.1.2.58
identity
distinguishing character or personality of an individual or entity

3.1.2.59
independent key
<information security> symmetric key that is derived from a high entropy bit source and not from a prior key

3.1.2.60
infrastructure
technical structures that support data communications within a facility

EXAMPLE Parts of a plant's IT network, perhaps using IEEE 802.3 or IEEE 802.11, or IEC 61158 Type 10 (PROFInet) or IEC 61158 Type 9 (FOUNDATION™ Fieldbus HSE).[3]

[3] PROFInet and FOUNDATION Fieldbus are the trademarks of various trade organizations. This information is given for the convenience of users of the standard and does not constitute an endorsement of the trademark holders or any of their products. Compliance to this profile does not require use of the registered trademark. Use of the trademarks requires permission of the trade name holder.

3.1.2.61
initialization vector
<information security> block of bits that is required to allow a cryptographic cipher in a streaming mode of operation to produce a unique stream independent from other streams produced under the same encryption key

3.1.2.62
interoperable
able to work together to perform a specific role in one or more distributed application programs

Note 1 to entry: In this case parameters and their application-related functionality fit together both syntactically and semantically. Interoperability is achieved when the devices support complementary sets of parameters and functions belonging to the same profile.

[SOURCE: IEC TR 62390:2005, 6.2.2.6, modified – adapted.]

3.1.2.63
interworkable
able to transfer parameters among correspondents

Note 1 to entry: In addition to the communication protocol, communication interface and data access, the parameter data types are the same.

[SOURCE: IEC TR 62390:2005, 6.2.2.5, modified – adapted.]

3.1.2.64
kerberos protocol
specific network authentication protocol that allows individuals communicating over an insecure network to prove their identity to one another in a secure manner

3.1.2.65
key agreement
<information security> process of establishing a shared secret key between entities in such a way that neither of them can predetermine the value of that key

[SOURCE: ISO/IEC 11770-1:2010, 2.13]

3.1.2.66
key archive
key management archive
<information security> encryption system with a backup decryption capability that allows authorized persons, under certain prescribed conditions, to decrypt ciphertext with the help of information supplied by one or more trusted parties which hold special data recovery keys

3.1.2.67
key center
<information security> centralized key distribution process, usually a separate computer system, that uses key-encrypting keys (master keys) to encrypt and distribute T-keys needed by a community of users

Note 1 to entry: Key centers generally are certified, traceable to an accredited independent testing agency, as meeting the requirements of ISO/IEC 19790 (similar to FIPS 140-2) for a Level 3 or Level 4 cryptographic module.

3.1.2.68
key confirmation
<information security> assurance for one entity that another identified entity is in possession of the correct key

[SOURCE: ISO/IEC 11770-1:2010, 2.16]

3.1.2.69
key de-registration
<information security> marking of all keying material records and associations to indicate that the key is no longer in use

3.1.2.70
key derivation
<information security> process by which one or more keys are derived from a shared secret and other information

3.1.2.71
key distribution
<information security> transport of a key and other keying material from an entity that either owns the key or generates the key to another entity that is intended to use the key

3.1.2.72
key distribution center
entity that is trusted to generate or acquire keys and to distribute the keys to communicating parties and that shares a unique symmetric key with each of the parties

[SOURCE: ISO/IEC 11770-1:2010, 2.22]

3.1.2.73
key encrypting key
<information security> cryptographic key that is used for the encryption or decryption of other keys

Note 1 to entry: Best practice is to limit use of symmetric (secret) KEKs to key wrapping and not use them for key transport or session (i.e., data) keys.

Note 2 to entry: KEKs may form a hierarchy. In this standard KEKs are often referred to as “master keys”, although the two concepts are not synonymous.

3.1.2.74
key escrow
<information security> process of recording keys and any related essential key recovery information in a key archive

3.1.2.75
key establishment
<information security> process by which cryptographic keys are securely distributed among cryptographic modules using manual transport methods (e.g., key loaders), automated methods (e.g., key transport and/or key agreement protocols), or a combination of automated and manual methods (consisting of key transport plus key agreement)

3.1.2.76
keying material installation
<information security> installation of keying material for operational use

3.1.2.77
key management

<information security> administration and use of generation, registration, certification, deregistration, distribution, installation, storage, archiving, revocation, derivation and destruction of keying material in accordance with a security policy

[SOURCE: ISO/IEC 11770-1:2010, 2.28]

3.1.2.78
key management infrastructure
<information security> framework and services that provide for the generation, production, distribution, control, accounting, and destruction of all cryptographic material, including symmetric keys, as well as public key signing and generation of its own static and ephemeral asymmetric-key pairs

Note 1 to entry: This includes all elements (hardware, software, other equipment, and documentation); facilities; personnel; procedures; standards; and information products that form the system that distributes, manages, and supports the delivery of cryptographic products and services to end users.

Note 2 to entry: Key management services include key ordering, distribution, re-key, update of keying material attributes, certificate revocation, key recovery and the distribution, accounting, tracking, and control of software that performs either keying material security or cryptographic functions.

3.1.2.79
key manager
<information security> key management infrastructure device that provides key management services

3.1.2.80
key pair
asymmetric key pair
<information security> pair of related keys where the private key defines the private transformation and the public key defines the public transformation

Note 1 to entry: A key pair is used with asymmetric-key cryptographic algorithms.

[SOURCE: ISO/IEC 11770-1:2010, 2.2]

3.1.2.81
key recovery
<information security> mechanisms and processes that allow authorized entities to retrieve keying material from secure key backup or archive storage

3.1.2.82
key registration
<information security> process of officially recording the keying material by a registration authority

3.1.2.83
key revocation
<information security> process whereby a notice is made available to affected entities that keying material should be removed from operational use prior to the end of the established cryptoperiod of that keying material

3.1.2.84
key transport
<information security> process of transferring a key from one entity to another entity, suitably protected

Note 1 to entry: When used in conjunction with a public key (asymmetric) algorithm, the keying material is encrypted using the public key of the receiver and subsequently decrypted using the private key of the receiver. When used in conjunction with a symmetric algorithm, the keying material is wrapped with a key encrypting key shared by the two parties.

[SOURCE: ISO/IEC 11770-1:2010, 2.33]

3.1.2.85
key update
<information security> function performed on a cryptographic key in order to compute a new but related key

3.1.2.86
key usage period
<information security> either the originator usage period or the recipient usage period of a symmetric key

3.1.2.87
key wrapping
<information security> method of encrypting keys (along with associated integrity information) that provides both confidentiality and integrity protection using a symmetric key

3.1.2.88
key wrapping key
<information security> (symmetric-key) key encryption key

3.1.2.89
keying material
<information security> data necessary to establish and maintain cryptographic keying relationships

EXAMPLE Keys, initialization values, periods of validity.

[SOURCE: ISO/IEC 11770-1:2010, 2.27]

3.1.2.90
latency
delay from when data is created at a data source device to when it is available to be consumed at the destination device

Note 1 to entry: The designated points of measurement are a) physical devices, or b) layer boundaries within multi-layer software (e.g., from sending transport to receiving transport functionality, or from sending application to sending modem).

3.1.2.91
lease
per-session fine-grained communication resource allocation occurring at a GIAP

3.1.2.92
least privilege
<information security> security principle that restricts the access privileges (e.g., program execution privileges, file modification privileges) of authorized personnel and their cyber agents to the minimum necessary to perform their jobs

3.1.2.93
link
momentary or persistent interconnecting path between two or more devices for the purpose of transmitting and receiving messaging

3.1.2.94
master key
<information security> cryptographic key that is used for deriving other keys

Note 1 to entry: Best practice prohibits using master keys as session (i.e., data) keys, which would ease their cryptanalysis. They may be used as KEKs, often at the top of a KEK hierarchy.

3.1.2.95
mesh topology
network topology in which redundant physically-diverse routing paths are available between each pair of network nodes

Note 1 to entry: Wireless mesh topology is usable to extend coverage via multi-hop capability and/or to facilitate communication reliability by providing redundant paths between devices.

3.1.2.96
message authentication
PDU authentication
<information security> process of establishing that a message was formed by a member of an authorized group of communicants and that the message is unchanged since it was formed

3.1.2.97
message authentication code
message integrity code
<information security> cryptographic checksum generated using a symmetric key that is typically appended to data in order to provide data integrity and source authentication similar to a digital signature

[SOURCE: ISO/IEC 26907:2009, 4.16]

3.1.2.98
message authentication code algorithm
<information security> algorithm for computing a function which maps strings of bits and a secret key to fixed-size strings of bits, satisfying the following two properties:

- for any key and any input string, the function can be computed efficiently;
- for any fixed key, and given no prior knowledge of the key, it is computationally infeasible to compute the function value on any new input string, even given knowledge of a set of input strings and corresponding function values, where the value of the ith input string might have been chosen after observing the value of the first i-1 function values (for integers i> 1)

[SOURCE: ISO/IEC 9797-1:2011, 3.10, modified – deletion of notes judged not relevant to this standard.]

3.1.2.99
MIC-computation syntax
<information security> concrete representation of an N-SDU associated protocol information, usually added via a prefix pseudo-header, that is used to bind selective N-addresses and N-PCI, and sometimes selective (N-1)-addresses and (N-1)-PCI, to the N-SDU before computing an integrity check code (MIC) over the assemblage

3.1.2.100
multicast
messaging from a source to a set of intended recipients

Note 1 to entry: The set membership is either indeterminate or determinate, where the latter includes the null set.

Note 2 to entry: Broadcast is a special form of multicast, usually to an indeterminate set of intended recipients. See also unicast.

Note 3 to entry: Multicast, other than broadcast, is not supported in this standard.

3.1.2.101
network management object
application software within a device that acts as a local peer of a network manager

3.1.2.102
non-deterministic bit generator
<information security> random bit generator whose security depends upon sampling an entropy source

Note 1 to entry: Sources of such bits include avalanche breakdown of a Zener diode, shot noise, thermal noise, radioactive decay, cosmic rays, etc.

Note 2 to entry: Post-processing of such noise sources is required to whiten their output and to detect failures in the circuit providing the randomness. Only the post-processed bit stream is suitable for seeding a deterministic bit generator.

[SOURCE: ISO/IEC 18031:2011, 3.23, modified – the original note to entry has been deleted and new notes to entry added.]

3.1.2.103
non-repudiation
<information security> ability to prove the occurrence of a claimed event or action and its originating entities, in order to resolve disputes about the occurrence or non-occurrence of the event or action and involvement of entities in the event

Note 1 to entry: In a general information security context, non-repudiation provides assurance that the originator of information is provided with durable proof of delivery or the recipient is provided with durable proof of the originator's identity, so that the party that provided the non-repudiable proof has no credible later denial of having processed the information.

[SOURCE: ISO/IEC 27000:2014, 2.54, modified – the "in order to ..." purpose clause was added.]

3.1.2.104
nonce
<information security> number used once, or a value that has (at most) a negligible chance of repeating

3.1.2.105
operational phase
operational use
<information security> phase in the lifecycle of keying material whereby the keying material is used for standard cryptographic purposes

3.1.2.106
operational storage
<information security> normal storage of operational keying material during its cryptoperiod

3.1.2.107
originator usage period
<information security> period of time during the cryptoperiod of a symmetric key during which cryptographic protection may be applied to data

3.1.2.108
period of protection
<information security> period of time during which the integrity and/or confidentiality of a key needs to be maintained

3.1.2.109
plaintext
<information security> unencrypted information (relative to an encryption or decryption process)

Note 1 to entry: Usually, the plaintext input to an encryption operation is not already enciphered, but in some cases the input is itself the output of another cryptographic operation.

[SOURCE: ISO/IEC 10116:2006, 3.11, modified by parenthetical comment.]

3.1.2.110
policy-based management
administrative (managerial) approach used to simplify the management of a given system via the establishment of policies in order to deal with situations that are understood to be likely to occur

3.1.2.111
private key
<information security> (cryptographic) key of an entity's asymmetric-key pair that is kept private

Note 1 to entry: The security of an asymmetric system depends on the privacy of this key.

Note 2 to entry: In an asymmetric (public) cryptosystem, the private key is associated with a public key. The private key is known only by the owner of the key pair and is used to:

– compute the corresponding public key;

– compute a digital signature that is verifiable by the corresponding public key;

– decrypt data that was encrypted by the corresponding public key; or

– compute a piece of common shared data, together with other information.

[SOURCE: ISO/IEC 11770-1:2010, 2.35, modified – a second note to entry has been added.]

3.1.2.112
pseudo-header
information that is logically prepended to a PDU before computing a MIC for the PDU, but which is not explicitly conveyed by the PDU

3.1.2.113
public key
<information security> key of an entity's asymmetric-key pair which can usually be made public without compromising security

[SOURCE: ISO/IEC 11770-1:2010, 2.36]

3.1.2.114
public key certificate
<information security> public key information of an entity signed by the certification authority

Note 1 to entry: Additional information in the certificate is able to specify how the key is used and its cryptoperiod.

[SOURCE: ISO/IEC 11770-1:2010, 2.37, modified – a note to entry has been added.]

3.1.2.115
recipient usage period
<information security> period of time during the cryptoperiod of a symmetric key during which the protected information is processed

Note 1 to entry: This period frequently extends beyond the originator's period of permitted usage.

3.1.2.116
resilience
ability of a functional unit to continue to perform a required function in the presence of faults or errors

[SOURCE: ISO/IEC 2382-14:1997, 14.04.06]

3.1.2.117
retention period
<information security> minimum amount of time that a key or other cryptographic related information should be retained in an archive

3.1.2.118
robustness
degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions

[SOURCE: ISO/IEC/IEEE 24765:2010, 3.2601]

3.1.2.119
router
device that forwards NPDUs within a computer network based on network-layer information

3.1.2.120
short control signaling
short MAC messaging, sent by a nominal receiver of MAC messaging as an immediate response to the originator of that MAC messaging, used to send control information such as ARQ ACK/NAK status, received signal quality and level, etc., as a way of informing the originating device of the reception status and of instantaneous conditions on the medium

Note 1 to entry: This note applies to the French language only.

3.1.2.121
secret key
<information security> key used with symmetric cryptographic techniques by a specified set of entities

[SOURCE: ISO/IEC 11770-3:2008, 3.35]

3.1.2.122
secure communications protocol
<information security> communication protocol that provides the appropriate confidentiality, authentication, content integrity and message timing protection

3.1.2.123
security association
<information security> relationship between two or more entities for which there exist attributes (state information and rules) to govern the provision of security services involving those entities

[SOURCE: ISO/IEC 10745:1995, 3.8]

3.1.2.124
security domain
<information security> set of assets and resources subject to a common security policy

Note 1 to entry: Security domains often are organized (e.g., hierarchically) to form larger domains.

[SOURCE: ISO/IEC 18028-3:2005, 3.19, modified – a note to entry has been added.]

3.1.2.125
security manager
<information security> application software that supervises various operational security aspects of a multi-device network, usually through interaction with device security management objects (DSMO) in the supervised device(s)

Note 1 to entry: A network security manager often is a dedicated device that is protected both physically and by construction. (See ISO/IEC 19790 (similar to FIPS 140-2) and the NIST/CSE Cryptographic Module Validation Program, http://csrc.nist.gov/cryptval/cmvp.htm.)

3.1.2.126
security services
<information security> mechanisms used to provide confidentiality, data integrity, authentication and/or non-repudiation of information

3.1.2.127
separation of duties
<information security> security principle that divides critical functions among different staff members in an attempt to ensure that no one individual has enough information or access privilege to perpetrate damaging fraud

3.1.2.128
session
T-association

3.1.2.129
session key
T-key
temporary data key used by TLEs

3.1.2.130
signature generation
<information security> use of a digital signature algorithm and a private key to generate a digital signature on data

3.1.2.131
signature authentication
<information security> use of a digital signature algorithm and a public key to verify a digital signature on data

3.1.2.132
source authentication
<information security> process of corroborating that the source of data is as claimed

3.1.2.133
split knowledge
<information security> process by which a cryptographic key is split into multiple key components, individually disclosing no knowledge of the original key other than possibly its size, which subsequently can be combined in any of a number of predefined groupings of the multiple specific keys to recreate the original key

3.1.2.134
static key
<information security> key that is not an ephemeral key, which is intended for use for a relatively long period of time, typically in successive invocations of a cryptographic key establishment scheme

3.1.2.135
stream cipher
<information security> cryptographic primitive that uses a symmetric key and an initialization vector and that works with continuous streams of input rather than fixed blocks

3.1.2.136
subnet
(N)-subnet
D-subnet
N-subnet
sub-network, a subset of a full network, either at data-link or network layer (layers 2 or 3 of the OSI Basic Reference Model), comprised of multiple end-point and relay nodes that are interconnected via a frequently-homogeneous (N-1)-layer

3.1.2.137
superframe
collection of timeslots with a common repetition period and possibly other common attributes

3.1.2.138
symmetric key
<information security> secret key shared between two or more parties that may be used for both encryption and decryption as well as for message integrity code computation and verification

[SOURCE: ISO/IEC 26907:2009, 4.27]

3.1.2.139
symmetric-key algorithm
symmetric-key (cryptographic) algorithm
<information security> cryptographic algorithm that uses the same (usually secret) key for an operation and its inverse (e.g., encryption and decryption)

3.1.2.140
system initialization
<information security> setting up and configuring a system for secure operation

3.1.2.141
system manager
application software that supervises various operational aspects of a multi-device network other than security, usually through interaction with network management objects in the supervised device(s)

Note 1 to entry: A network manager either supervises an entire multi-device network or acts as a subordinate to another network manager, thereby supervising only a subset of the entire network.

Note 2 to entry: A network manager is not always a dedicated device.

3.1.2.142
system/security manager
system manager functionality acting on behalf of the security manager functionality

3.1.2.143
threat
<information security> circumstance or event with the potential to adversely impact plant operations (including function, image, or reputation), assets or individuals through an information system via unauthorized access, destruction, disclosure, modification of data, message delay, and/or denial of service

3.1.2.144
threshhold
value that when exceeded in a designated direction results in a specified reaction

3.1.2.145
time window
interval of time

3.1.2.146
D-transaction
MPDU that is not an immediate acknowledgment MPDU, plus any sequence of zero or more immediate acknowledgment MPDUs that immediately follow and are a consequence of the first MPDU (and optionally each other), all on the same channel and in the same time slot

3.1.2.147
tunneling
encapsulation of a first protocol within a second communication protocol to convey PDUs from the first protocol

3.1.2.148
type-of-service
collective name given to a set of protocol elements and associated quality-of-service attributes that together form a subprotocol (e.g., real-time voice, time-critical data, and non-time-critical data) with distinct functionality

3.1.2.149
unauthorized disclosure
<information security> event involving the exposure of information to entities not authorized access to the information

3.1.2.150
unicast
messaging from a source to a single intended recipient

Note 1 to entry: See also multicast and broadcast.

3.1.2.151
user initialization
process whereby a user prepares the cryptographic application for use (e.g., installing and configuring software and hardware)

3.1.2.152
user registration
process whereby an entity becomes a member of a security domain

3.1.2.153
zeroisation
<information security> method of erasing electronically stored data, cryptographic keys, and critical stored parameters by altering or deleting the contents of storage in a manner that prevents recovery of the data

3.1.3 Symbols for symmetric keys, and for asymmetric keys and certificates

NOTE Symmetric keys defined by this standard are 128-bit keys. Asymmetric keys defined by this standard have a similar hypothesized cryptographic bit strength of 128 bits or greater. See Clause 7 for the exact description of the cryptographic material.

3.1.3.1
K_DL
current data key for all devices in the local D-subnet

3.1.3.2
K_global
well-known key whose value is static and published, which is used to provide uniformity of PDU structure and processing when a shared-secret key is inappropriate or unknown

3.1.3.3
K_open
well-known limited-utility key, different from K_global, whose value is static and published, which can be used as the value of K_join to provision a device via its over-the-air interface

Note 1 to entry: Use of this key in an environment where eavesdropping could occur can compromise the security of the device and its relationships to the rest of the wireless system.

3.1.3.4
K_join
key used to bootstrap a new device securely into the network

3.1.3.5
K_join_wrapped
wrapped version of the join key, used to recover from a failed security manager

3.1.3.6
K_master
master key used as a KEK for key distribution and security management of a single device

3.1.3.7
K_session_AB
current data key for a session between device A and device B, identical with K_session_BA

3.1.3.8
CA_root
public key of a certificate authority which signed a device's public-key certificate

Note 1 to entry: This key is commonly referred to as a root key and is used to assist in verifying the true identity of the device communicating the certificate, as well as some related keying information.

3.1.3.9
Cert-A
public-key certificate of device A, used to evidence the true identity of the device, as well as related keying information, during execution of an authenticated public-key key establishment protocol

3.1.4 Terms used to describe device behavior

3.1.4.1
capability
ability to perform actions, including attributes on qualifications and measures of the ability as capacity

EXAMPLE The number of connected devices that a router can support.

Note 1 to entry: Profiles specify a minimum capability.

[SOURCE: ISO 18435-2:2012, 3.6, modified – addition of the example and note]

3.1.4.2
configuration
set of parameters that 1) alter behavior and 2) can be set by a system manager

EXAMPLE Network-layer hop limit.

Note 1 to entry: Configurations where defaults are appropriate state those defaults (e.g., NL hop limit = 64).

3.1.4.3
configure
act of specifying and administering configuration parameters

3.1.4.4
feature
notable characteristic of a device

EXAMPLE Battery-powered.

3.1.4.5
mandatory
required element for any claim of compliance to this standard

EXAMPLE Support for symmetric-key cryptography.

3.1.4.6
optional
element that is not required to claim conformance with this standard but which, if present, is required to behave as specified in this standard

EXAMPLE Support for asymmetric-key cryptography.

3.2 Abbreviated terms and acronyms

6LoWPAN	IPv6 over a low power personal area network (PAN)
6TSCH	time-slotted channel hopping (TSCH) with 6LoWPAN
ACK	positive acknowledgment
AE	application-entity
AES	advanced encryption standard (see ISO/IEC 18033-3)
AID	attribute ID
AL	application layer
ALE	application layer entity
AME	application layer management entity
APDU	application layer protocol data unit
ARMO	alert reporting management object
ARO	alert-receiving object
ARQ	automatic repeat request
ASAP	application layer service access point
ASDU	application layer service data unit
ASL	application sublayer
ASLMO	application sublayer management object
ASMSAP	application sublayer management service access point
ASM	asset management
ASN.1	abstract syntax notation one
ATT	address translation table
AUTO	automatic mode (of an automation process)
BBR	backbone router
BECN	backward explicit congestion notification

BRPT	backbone router peer table
C/S	client/server
CBC	cipher-block-chaining stream cipher mode (see NIST SP 800-38C)
CCA	clear channel assessment
CCM	counter with cipher-block-chaining message authentication mode
CCM*	CCM enhanced
CIP	Common Industrial Protocol™[4]
CON	concentrator object
CoS	class of service
CoSt	change of state
CSMA	carrier sense multiple access
CSMA/CA	carrier sense multiple access with collision avoidance
dBm	dB (1 mW)
DADDR	data-link address
DAUX	data-link auxiliary header
DBP	device being provisioned
DCS	distributed control system
DD	device description
DDE	data-link layer data (sub-)entity
DDL	device description language
DDSAP	data-link layer data service access point
DE	discard eligible
DIS	dispersion object
DK	(symmetric) data key, data authentication key, data encryption key
DL	data-link layer
DLE	data-link layer entity
DLMO	data-link layer management object
DMAP	device management application process
DME	data-link layer management (sub-)entity
DMIC	data-link layer message integrity code
DMO	device management object
DMSAP	data-link layer management service access point
DMSO	device management service object
DMXHR	data-link layer management extension header
DPDU	data-link layer protocol data unit
DPO	device provisioning object
DPSO	device provisioning service object
DROUT	data-link layer routing (subheader)
DSAP	data-link layer service access point
DSC	data-link layer security component
DSDU	data-link layer service data unit

[4] Property of ODVA, http://www.odva.org.

DSMO	device security management object
DSO	directory service object
DSSS	direct sequence spread spectrum
DWT	data-link layer protocol data unit wait time
EC	European Community
ECB	electronic code book
ECC	elliptic curve cryptography standard (see ISO/IEC 18033-2)
ECMQV	Menezes-Qu-Vanstone algorithms using elliptic curve cryptography
ECN	explicit congestion notification
ECPVS	Pintsov-Vanstone algorithms using elliptic curve cryptography
ECQV	Qu-Vanstone algorithms using elliptic curve cryptography
ED	energy detection
EDDL	electronic device description language
EIRP	effective isotropic radiated power
EMA	exponential moving average
ETSI	European Telecommunications Standards Institute
EUI-64™[5]	64-bit extended unique identifier as specified by the IEEE
FDT/DTM	field device tool / device type manager
FCC	U.S. Federal Communications Commission
FCS	frame check sequence
FEC	forward error correction
FECN	forward explicit congestion notification
FF-H1	Foundation Fieldbus – H1 protocol
FF-HSE	Foundation Fieldbus – high speed ethernet
FHSS	frequency hopping spread spectrum
FHSSM	frequency hopping spread spectrum modulation
FIFO	first in, first out (queuing discipline)
FIPS	[US] federal information processing standard (issued by NIST)
FPAC	foreign protocol application communication
FPT	foreign protocol translator
FPT-PAI	foreign protocol translator – protocol address information
FPT-PCI	foreign protocol translator – protocol control information
FPT-PDU	foreign protocol translator – protocol data unit
FSK	frequency shift keying
GIAP	gateway interface access point
GPS	global positioning system
GUC	number of supportable gateway-UAP connections
GUI	graphical user interface
HART	highway addressable remote transducer
HC	header compression
HCF	HART Communication Foundation

[5] Property of the trademark owner.

HMAC	(keyed)-hash message authentication code
HMI	human-machine interface
HRCO	health reports concentrator object
(N)-ICI	(N-layer) interface control information
ICMP	internet control messaging protocol
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical & Electronics Engineers
IFO	interface object
INF	infinity
I/O	input/output
IPv4	internet protocol version 4
IPv6	internet protocol version 6
ISA	International Society of Automation
ISM	industrial, scientific, medical
IV	initialization vector
JT_n	join timer n
KDC	key distribution center
KEK	key encryption key
LAN	local area network
LBT	listen before talk
LH	last hop
LLC	logical link control sublayer (in upper DL)
LP	low power
LQI	link quality indicator
LSB	least significant bit
MAC	media access control sublayer (spanning lower DL and upper PhL)
MAN	manual mode (of an automation process)
MIB	management information base
MIC	message integrity code
MICI	media access control interface control information
MK	(symmetric) master key
MP	management process
MPCI	media access control sublayer protocol control information
MPDU	media access control sublayer protocol data unit
MSB	most significant bit
NAK	negative acknowledgment
NaN	not-a-number
NDE	network layer data (sub-)entity
NDSAP	network layer data service access point
NFC	near-field communications
NICI	network interface control information
NIDS	network intrusion detection system
NIST	[US] National Institute of Standards and Technology

NL	network layer
NLE	network layer entity
NME	network layer management (sub-)entity
NMSAP	network layer management service access point
NO	native object
NPCI	network layer protocol control information
NPDU	network layer protocol data unit
NPDU.F128	final destination IPv6Address in the expanded network header
NPDU.F16	final destination DL16Address in the expanded network header
NPDU.O128	originator IPv6Address in the expanded network header
NPDU.O16	originator DL16Address in the expanded network header
NSAP	network layer service access point
NSD	number of system devices
NSDU	network layer service data unit
OBJ	generic application layer object
ODVA	Open Device Vendor Association (now legally known only by its acronym)
OOB	out-of-band
OOS	out of service mode (of an automation process)
OP	(automation process) output point
OPC	open connectivity in industrial automation
OSI	Open Systems Interconnection
OTA	over-the-air
P/S	publish/subscribe
PA	process automation
(N)-PAI	(N-layer) protocol addressing information
PAN	personal area network
PCH	PHY coding header
(N)-PCI	(N-layer) protocol control information
PD	provisioning device
(N)-PDU	(N-layer) protocol data unit
PhD	physical layer data service
PhICI	physical layer interface control information
PhL	physical layer
PhLE	physical layer entity
PhPDU	physical layer protocol data unit
PhSAP	physical layer service access point
PhSDU	physical layer service data unit
PHY	physical layer (as used in IEEE 802 standards)
PIB	policy information base
PICS	protocol implementation conformance statement
PKI	public key infrastructure
PNO	PROFIBUS Nutzerorganisation (PROFIBUS User Organization)
PSH	PHY synchronization header

PSMO	proxy security management object
PV	(automation process) process variable
PWT	PDU wait time
QoS	quality of service
R&TTE	radio and telecommunications terminal equipment
RDP	reliable datagram protocol
RF	radio frequency
RFC	request for comments
RFP	request for proposal
RSSI	received signal strength indicator
RSQI	received signal quality indicator
RT	routing table
RTO	retry time-out interval
RTT	round-trip time
RTU	remote terminal unit
RTTV	round-trip time variation
S/S	source/sink
(N)-SAP	(N-layer) service access point
SCADA	supervisory control and data acquisition
SCS	short control signaling
SCO	system communication configuration object
(N)-SDU	(N-layer) service data unit
SFD	start frame delimiter
SIFS	short inter-frame separation
(U)SIM	(universal) subscriber identity module
SINR	signal to interference plus noise ratio
SK	(symmetric) T-key used for authentication and confidentiality
SKG	secret key generation
SL	session layer
SMIB	structured management information base
SMAP	system manager application process
SM	system manager
SMAP	system management application process
SMO	system monitoring object
SOE	sequence of events
SRTT	smoothed round-trip time
STSO	system time service object
TAI	international atomic time; temps atomique international
TCP	transmission control protocol
TDMA	time division multiple access
TDE	transport layer data (sub-)entity
TDSAP	transport layer data service access point
TFRC	TCP-friendly rate control, IETF RFC 5348

TICI	transport interface control information
TL	transport layer
TLE	transport layer entity
TME	transport layer management (sub-)entity
TMIB	transport layer management information base
TMIC	transport layer message integrity code
TMSAP	transport layer management service access point
ToS	type of service
TPCI	transport layer protocol control information
TPDU	transport layer protocol data unit
TSAP	transport layer service access point
TSC	transport layer security component
TSCH	time-slotted channel hopping
TSDU	transport layer service data unit
TUN	tunnel object
TUN-Data	tunnel data
UAL	upper application layer
UAP	user application process
UAPMO	user application process management object
UDO	upload/download object
UDP	user datagram protocol
UFO	unified field object
UTC	universal coordinated time; temps universel coordonné
WBM	wideband modulation
WiMAX	worldwide interoperability for microwave access
WISN	wireless industrial sensor (and actuator) network

3.3 Conventions

3.3.1 Service interfaces

Portions of this standard use the descriptive conventions given in ISO/IEC 10731.

Service primitives, used to represent service user/service provider interactions (see ISO/IEC 10731), convey parameters that indicate information available in the user/provider interaction.

This standard uses a tabular format to describe the component parameters of the NS (N layer SAP or entity SAP) primitives. The parameters that apply to each group of NS primitives are set out in tables. Each table consists of up to six columns, containing the name of the service parameter, and a column each for those primitives and parameter-transfer directions used by the NS:

- the request primitive's input parameters;
- the indication primitive's output parameters;
- the response primitive's input parameters; and
- the confirm primitive's output parameters.

NOTE 1 The request, indication, response, and confirm primitives are also known as requestor.submit, acceptor.deliver, acceptor.submit, and requestor.deliver primitives, respectively (see ISO/IEC 10731).

One parameter (or part of it) is listed in each row of each table. Under the appropriate service primitive columns, a code is used to specify the type of usage of the parameter on the primitive and parameter direction specified in the column:

M: parameter is mandatory for the primitive;

S: selection from a defined set of two or more parameters;

U: parameter is a user option and may or may not be provided depending on the dynamic usage of the NS-user. When not provided, a default value for the parameter is assumed;

C: parameter is conditional upon other parameters or upon the environment of the NS-user;

(blank) or em-dash ("—"): parameter is never present.

Some entries are further qualified by items in brackets. These may be:

- a parameter-specific constraint:

(=) indicates that the parameter is semantically equivalent to the parameter in the service primitive to its immediate left in the table;

- an indication that some note applies to the entry:

(n) indicates that the following note n contains additional information pertaining to the parameter and its use. Letter-enumerated notes are normative; digit-numbered notes are informative.

A summary table is provided to define the parameter usage within the primitive. Each cell defines whether each parameter is mandatory, optional, prohibited, or conditional.

The ordering of the parameters is also implicitly defined from top to bottom.

Complex parameters may also be shown. For example, a structure may be mandatory, but some of the elements of the structure may be optional. The structure elements are indented proportional to their level of hierarchy.

Input parameters for services are specified for request and response service primitives. Output parameters for services are specified for indication and confirmation service primitives.

The following abbreviations are used in the service tables:

- Request service request
- Indication service indication
- Response service response
- Confirmation service confirmation

NOTE 2 Intra-device handling of inter-layer error situations, such as situations where a lower layer queue overflows or a lower layer timeout occurs, is a local matter and hence is not addressed by this standard.

3.3.2 Table cells

For all tables, table entries that are irrelevant or unspecified may contain an em-dash ("—"). Table entries to be filled by suppliers may contain an ellipsis ("…").

3.3.3 Italics

In some cases a generic term that is used for a specific purpose is italicized at first occurrence, to indicate to the reader that care in interpretation is suggested. The reasons for other uses of italics, such as in 4.5.5.2.1, should be apparent from their immediate context.

3.3.4 Bold face

In some cases a descriptive paragraph is preceded by a bold-faced term or summarizing descriptive phrase, where the bold-facing is used to assist the reader's future memory.

NOTE Such use of bold-face is inessential, so loss of such distinction due to photocopying is not important.

3.3.5 Informal declarations of named constants

ASN.1 permits numeric constants to be assigned symbolic names. It also permits named constants of enumerations to be assigned specific numeric representations. In this standard the two are unified, when used as inline declarations of permissible choices for fields of data structures, through use of the syntax

numericValue ": " explanatory text

which is intended as the equivalent to the ASN.1 declaration

explanatory_text "(" numericValue ")"

where the explanatory text is converted into an alphanumeric identifier by replacing the spaces between words (if any) with underscores, and by adjusting any delimiting period, comma, or semicolon after the explanatory text to the required ASN.1 list element separator after the equivalent closing parenthesis of the numericValue.

4 Overview

4.1 General

This standard uses the OSI layer description methodology (see Annex C) to define protocol suite specifications, in addition to specifications for the functions of security, management, gateway, and provisioning for an industrial wireless network. The protocol layers supported are the physical layer (PhL), data-link layer (DL), network layer (NL), transport layer (TL), and the application layer (AL).

NOTE 1 Although this standard uses the concept of protocol layers, compliance to this standard does not mandate that implementations partition function similarly to that implied by these layers. Inter-layer interfaces are generally not exposed in a product, and hence are not suitable for conformance testing.

The wireless network defined by this standard consists of wireless devices serving usage classes 1 through 5 (described in Annex C) for non-critical applications of fixed, portable, and moving devices.

References to a network compliant to this standard will hereafter be referred to as a wireless industrial sensor network (WISN), even when that network contains actuators and other devices that are not logically classifiable as sensors.

Most devices that participate in a WISN are expected to implement just a single wireless PhLE and associated DLE. However, devices with multiple wireless PhLEs and associated DLEs are not precluded. Therefore this standard distinguishes between requirements for a device and requirements for a PhLE and associated DLE, even though the two are usually thought of as synonymous.

NOTE 2 It is suggested that the reader internalize this relationship, so that encountering the term DLE brings to mind the term device, even though in rare cases they are not one-to-one.

4.2 Interoperability and related issues

IEC TR 62390 provides useful definitions for differing levels of interoperation. The following three definitions, each of which includes the former, are quoted exactly from that technical report.

NOTE 1 IEC TR 62390:2005, Figure 9, provides additional clarity on the relationship of these terms.

NOTE 2 The notes in 4.2 were not in IEC TR 62390.

a) **Interconnectability**: Two or more devices are interconnectable if they are using the same communication protocols, communication interface and data access.

b) **Interworkability**: Two or more devices are interworkable if they can transfer parameters between them, i.e, in addition to the communication protocol, communication interface and data access, the parameter data types are the same.

NOTE 3 Interworkability implies interconnectability.

c) **Interoperability**: Two or more devices are interoperable if they can work together to perform a specific role in one or more distributed application programs. The parameters and their application-related functionality fit together both syntactically and semantically. Interoperability is achieved when the devices support complementary sets of parameters and functions belonging to the same profile.

NOTE 4 Interoperability implies interworkability, and thus also interconnectability.

4.3 Quality of service

To support multiple applications within a network along with diverse needs this standard supports multiple levels of quality of service (QoS). QoS describes parameters such as latency, throughput, and reliability. A device's application(s) request(s) the level of QoS needed. If the necessary resources can be made available for the requesting application, the system manager will allocate those resources in response to the requested QoS. See Clause 6 for additional information.

4.4 Worldwide applicability

This standard is intended to conform to established regulations in most world regions; however, its acceptability in any specific regulatory environment is not guaranteed and thus shall be evaluated. Annex V addresses this topic, including use in the EU under ETSI EN 300 328.

4.5 Network architecture

4.5.1 Interfaces

4.5.1.1 Defined interfaces

This standard defines service access points (SAPs) at the upper boundary of each protocol layer to decouple specifications of, and revisions to, each of those protocol layers from other layers, to the extent feasible. For example, if a new PhL is defined that does not require corresponding changes in the employing DL, then it can be added to the specification with minimal (if any) impact on the other protocol layers defined by this standard.

In most cases these defined interfaces are internal to an implementation. As such they are not subject to standardization and conformance testing, since such testing can be applied only to external interfaces of a unit under test (i.e., black box testing). As a consequence, such internal interfaces are descriptive and informative, not normative. However, it is expected that implementations which partition software along the lines suggested by these interfaces will be easier to maintain and adapt to future revisions of this standard.

Conformance to this standard applies only to the observable behavior of an implementation, including the structure and encoding of any information exchanged at observable interfaces that are specified as such by this standard.

4.5.1.2 Interfaces that are not defined

The following interfaces are not addressed by this standard.

- System manager to security manager: The security manager and the system manager form two core roles in the network that are closely related. Since the security manager and the system manager are so dependent upon each other, and because the security manager communicates directly only with the security manager, it is expected that the one or more devices that provide these two roles will be procured from a single vendor, often realized as a single device supporting both roles. Given these expectations, it is deemed not necessary to standardize this interface.

NOTE This interface is a subject of potential future standardization.

- External interfaces: An important attribute of this standard is that it is designed to allow the wireless network to leverage or integrate into a plant's communication infrastructure. This standard defines specific roles to allow this network to interface to other networks, including both wired and wireless networks and both standard and proprietary networks. However, since those specific external networks cannot be identified by this standard, neither can the interfaces to such networks be identified or specified.

4.5.2 Data structures

4.5.2.1 Defined PDUs

This standard defines the structure of protocol data units (PDUs) used for inter-device communication at the following protocol layers: PhL, DL, NL, TL and AL. Most of these are based on other international standards: ISO/IEC/IEEE standards for the PhL and DL, and IETF standards for the NL and TL. All such PDU definitions are normative, subject to external examination and conformance testing.

Conceptually, each distinct class of PDU has:

- an abstract transfer syntax, which describes the structure of the PDU, including order of fields and semantic meaning of each field and its alternative contents, and
- one or more concrete transfer syntaxes, which describe the encoding within the PDU for each of those fields and content alternatives.

This standard specifies a *full* or *canonical* concrete transfer syntax for each PDU, and for DPDUs, NPDUs and TPDUs also specifies a *compressed* concrete transfer syntax that reduces the energy requirements for PDU transmission and reception, as well as the occupancy time of the wireless channel when the PDU is being transmitted.

NOTE 1 Decreased channel occupancy reduces interference with other wireless devices and systems, as well as increasing the probability that the PDU is successfully received. It also reduces the average power required for device operation, which is of particular importance for devices not connected to an external power supply.

For DPDUs and TPDUs, a third concrete syntax (i.e., encoding) is employed when computing a message integrity code for the PDU. That encoding typically prepends a *pseudo-header* to the PDU as transmitted/received, where the pseudo-header contains specific lower-layer PDU addressing information, thus serving to bind that lower-layer information to the PDU whose integrity is covered by the computed MIC. In this standard this third concrete syntax is called the *MIC-computation syntax*.

NOTE 2 The PDU descriptions in this standard often conflate the abstract syntax of the PDU (i.e., its logical contents) with the concrete transfer syntax (i.e., its encoding). It is anticipated that a future edition of this standard will correct this deficiency.

4.5.2.2 Defined management data structures

This standard defines the structure of management data objects at various protocol layers. Such definitions are normative with respect to the behavior of defined operations on those data structures, and with respect to the presentation of those data structures to the extent, and in the form, that they occur when conveyed by PDUs. However, the representation of those data structures internal to an implementation is beyond the scope of standardization, since such representation is unobservable and thus not subject to conformance testing.

4.5.3 Network description

Figure 1 depicts the communication areas addressed by this standard, as well as those areas (shaded in blue) that are not within the scope of this standard. In Figure 1, circular objects represent roles for field devices (sensors, valves, actuators, etc.) and rectangular objects represent roles for infrastructure devices that communicate to other network devices via an interface to the network infrastructure backbone network.

NOTE This standard defines roles that devices embody; for further information on these roles, see 5.2.6.

A backbone is a data network (preferably high data rate) that is not defined by this standard. This backbone could be an industrial Ethernet, IEEE 802.11, or any other network within the facility interfacing to the plant's network. See Annex E for further information and assumptions about the characteristics of a backbone.

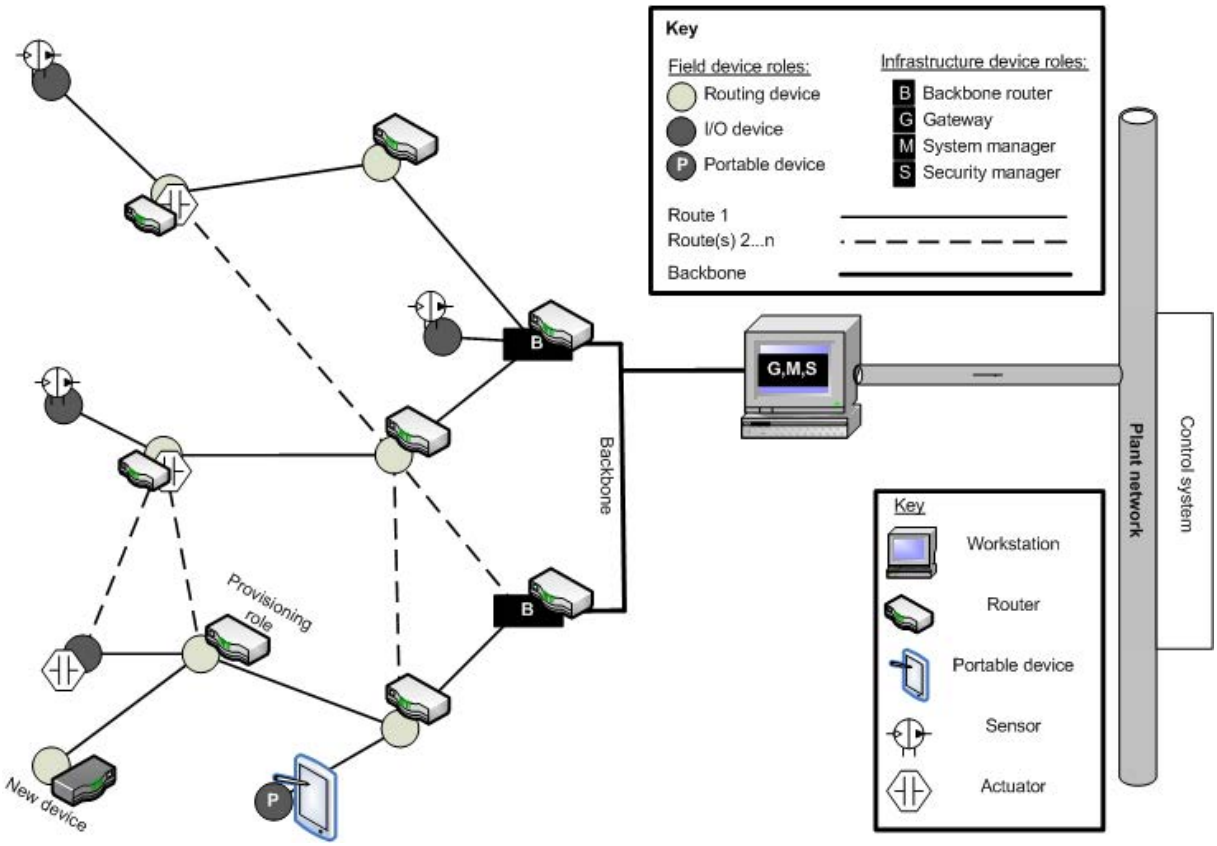


Figure 1 – Standard-compliant network

A complete network as defined in this standard includes all components and protocols required to route secure traffic, manage network resources, and integrate with host systems. A complete network consists of one or more field D-subnets that may be connected by an infrastructure device to a plant subnet.

A field D-subnet consists of a collection of field devices that wirelessly communicate using a protocol stack defined by this standard. As shown in Figure 1, some field devices may have routing capabilities, enabling them to forward messages from other devices.

A transit subnet consists of infrastructure devices on a backbone, such as backbone routers, gateways, system managers, and security managers. Since the backbone physical communication medium and its network protocol stack are outside the scope of this standard, they are not specified and may include tunneling compliant PDUs over external TL or AL protocols.

Devices that connect two disparate D-subnets have at least two DLE and PhLE interfaces. A backbone router connects a field D-subnet with a backbone D-subnet. A gateway connects a backbone subnet with a plant subnet; it may be collocated with a backbone router.

NOTE 1 The scope of a subnet depends on the uppermost communications layer used in construction of the subnet, which could be OSI layer 1, 2, 3 or 4.

NOTE 2 Since gateways are not defined in this standard, the nature of the interface that they provide to a "plant" network is strictly notional. Thus the term "plant network" in Figure 1 refers to whatever network or other communications means exists on the "far" side of the gateway, relative to the D-subnets that are covered by this standard. Similarly, the term "field subnet", where used, refers to the D-subnet composed directly of field devices.

All addressing, routing, and transport are limited to the scope of the field D-subnet. Each DLE within such a D-subnet is identified by a local DL16Address as well as an IPv6Address with global scope.

4.5.4 Generic protocol data unit construction

This communication standard uses communication protocol layers modeled in accord with the OSI Basic Reference Model. A protocol layer typically encapsulates the data it is conveying for a higher protocol layer, and in turn uses a lower layer to convey the encapsulated result.

The information conveyed across the network between peer entities is called a protocol data unit (PDU). The information conveyed at a layer boundary between layer entities of a single network node is called a service data unit (SDU). An SDU usually consists of a single PDU, but it can also be a group of concatenated PDUs (see concatenation, 3.1.1.16).

An SDU is considered to be an opaque octetstring or bitstring that is to be conveyed transparently (i.e., without interpretability or alteration) by the lower layer. The SDU can be conveyed by a single lower-layer protocol data unit (PDU), or be segmented (see segmenting, 3.1.1.58) to be conveyed piecemeal by many lower-layer PDUs, or be grouped with other SDUs (see blocking, 3.1.1.10) for conveyance as a group within a single lower-layer PDU. In the most common case, a header and footer are added to a single SDU to form a single protocol data unit (PDU), as shown in Figure 2.

NOTE The footer is usually either null or used for some form of message integrity code (MIC), such as an easily-spoofed computationally-simple checksum or a portion of a cryptographically-secured keyed hash.

The header and footer are often referred to as overhead with respect to the single or multiple or fractional conveyed SDU(s), with the amount of overhead depending upon how much additional information needs inclusion for the conveying protocol to function properly. Since one goal of this standard is to minimize energy consumption and channel occupancy when conveying PDUs, minimizing the amount of overhead at each protocol layer is a primary method of achieving that goal.

A complete description of each header and footer can be found in the appropriate protocol layer description. A full multi-layer PDU includes all headers and footers as shown in Figure 3. The amount of data (measured in octets) of an application PDU that can be sent in a single transmission is determined by the difference between the maximum permitted or supported PhL payload and the overhead imposed by all intermediary headers and footers.

Figure 2 – Typical single-layer PDU without fragmenting or blocking

Figure 3 – Full multi-layer PDU structure used by this standard

4.5.5 Abstract data and concrete representations

4.5.5.1 Abstract data types

Each protocol layer of this standard defines the structure of the PDUs that it exchanges with peer protocol entities at the same layer, and of the SDUs and related interface control information (ICI) that it exchanges with adjacent layer entities within a local node of the network.

Additionally, each protocol layer defines management data structures that are exchanged by each layer entity (as conveyed data) with remote systems management entities.

Each of these data structures has an abstract form that requires a concrete representation. The abstract elements are either scalars, or composites composed of scalars and other composites. Composites can be homogeneous, in which case they are single or multi-dimensional arrays (often known as a “vector” or “matrix”, respectively), or heterogeneous (often known simply as a “data structure”).

The scalar elements used by the protocol entities of this standard are:

a) integers of a constrained range, where each abstract value is represented by the equivalent two’s-complement or unsigned concrete binary value, depending on whether the abstract range includes negative values;

b) enumerations, usually declared as an UnsignedN representation for $N \leq 8$, where each abstract value generally is represented by the zero-origin ordinal index of the abstract value within the list of defined values;

c) Booleans, with two abstract values (FALSE, TRUE) to which the rules and operators of Boolean logic apply, with a concrete representation of FALSE as zero and TRUE as any non-zero binary value;

NOTE 1 Booleans are named after the logician George Boole.

NOTE 2 Although Booleans appear to be a special class of enumeration, the differences are that the value TRUE can be represented by any non-zero binary value, and that Boolean operators apply to this class.

d) IEEE floating point numbers of a specified range and precision, whose values are approximate real numbers or special non-numeric constants;

e) representations of TAI time as integer or scaled-fixed-point values modulo 2^{32} s.

The composite elements used by the protocol entities of this standard include three noteworthy classes of singly-dimensioned zero-origin arrays that are called *strings*. They are:

f) characters, known as *visible strings*;

g) bits, known as *bit strings*; and

h) uninterpreted octets, known as *octet strings*.

Other composite elements include packed Boolean arrays, which are often (incorrectly) conflated with their underlying representation as bit strings.

4.5.5.2 Declarations of abstract data elements and their concrete representations

4.5.5.2.1 Simple declarations

Within this standard, an abstract type and its concrete representation are often declared in a unified form that indicates both the class of 4.5.5.1 a) through h) and the number of bits in the underlying binary representation.

- Integers are declared with an implicit range as *unsignedN*, implying a range of $0..2^{N}-1$, or *signedN*, implying a range of $-2^{N-1}..2^{N-1}-1$, where N is the number of bits of the representation, usually a multiple of 8.

Integers that require a range other than that implied by their representation are declared as

unsignedN range min..max

where min and max are the minimum and maximum values of that integer element's range, respectively.

- Booleans are declared as *BooleanN*, where N is the number of bits of the representation, usually either 1 (when within a packed data structure) or 8.
- Floating point numbers and their range and precision are declared as *float32* or *float64*.
- Fixed-size strings and their sizes are declared as *visibleStringN*, *octetStringN* and *bitStringN*, where N is the number of elements in the underlying array.

NOTE 1 An internal coding mechanism within some visible strings, usually a null (0x00) used as a content-end delimiter, often is used to truncate the effective size of the contained string. Many libraries of string operators presume such a coding.

- Varying-size strings are declared as *visibleString* and *octetString* without a concatenated declared size (i.e., the N of *visibleStringN*).
- Packed Boolean arrays and their size are declared as *BooleanArrayN*, where N is the number of elements in the array (and hence the number of bits in the representation).
- Named constants, usually of values for UnsignedN fields, may be declared in ASN.1 fashion as if they were ASN.1 enumerations, as in

  UnsignedN {name-1(integer-value-1), … , name- K(integer-value-K)}

  where N is the number of bits of the representation, and an explicit declaration of the constants are provided in the form of a bracketed, comma-separated list, each element of which is followed by "(K)" where K is the value assigned to that element.

  EXAMPLE 1 A declaration for the protocol layers defined in this standard might be

  protocolLayers Unsigned3 {PhL(1), DL(2), NL(3), TL(4), AL(7)}

  NOTE 2 Example 1 demonstrates that the names of the elements of what amounts to an enumeration need not be disjoint from those used elsewhere in other places in this standard, as might be required for an explicit programming language specification.

  Alternatively, these named constants may be declared as

  UnsignedN {integer-value-1:description-1; … ; integer-value-K:description-K}

  where N is the number of bits of the representation, and an explicit declaration of the named constants in the form of a bracketed, semicolon-separated list, each element of which is preceded by "K:" where K is the value assigned to that named constant. (See 3.3.5.)

  EXAMPLE 2 A declaration for the join_method defined in this standard might be

  join_method Unsigned8{ 0:none; 1:join and start; 2:warm restart; 3:restart as provisioned; 4:reset to factory defaults }

  For this latter form of declaration, the bracketed list may be separated from the declaration of the field's representation.

  EXAMPLE 3 join_method Unsigned8

  Named constants:
  0:none
  1:join and start
  2:warm restart
  3:restart as provisioned
  4:reset to factory defaults

  NOTE 3 This latter form of declaration often occurs in tabular descriptions of data structures in this standard where the first part of the declaration is in one column and the second part is in the same or a different column of the same row.

Many times only a few elements of the range of an UnsignedN are named, such as may occur when the all-zero or all-ones value of the representation has a special interpretation. On some occasions, particularly when the description is “reserved”, a value range may be specified rather than just a single value.

4.5.5.2.2 Declarations of compound objects, and of methods and their arguments

Within this standard, compound data structures are usually declared in tables, each row of which (after heading rows) describes a constituent element within the data structure.

Within this standard, method descriptions are also usually declared in tables. Each such description specifies a method name, a numeric method ID and a method description, followed by a series of descriptions of method input arguments, followed by a series of descriptions of method output arguments. As with data structure definitions, each argument is declared in its own row of the table and has a declared type and, where relevant, a declaration of the alternatives of the associated named constants.

4.6 Network characteristics

4.6.1 General

Characteristics of a WISN (i.e., a wireless network that conforms to this standard) are:

- scalable;
- extensible;
- support for simple operation;
- license-exempt operation;
- robustness in the presence of interference and with non-WISNs;
- determinism or contention-free media access;
- self-organizing network with support for redundant communications from field device to plant network;

NOTE Redundancy support is not defined in this standard.

- IP-compatible network layer;
- coexistence with other wireless devices in the industrial workspace;
- security, including data authenticity, data confidentiality, data integrity, delay protection, and replay protection;
- system management of all communication devices;
- support for application processes using standard objects; and
- support for tunneling (i.e., transporting) other protocols through the wireless network.

4.6.2 Scalability

The architecture supports wireless systems that span the physical range from a single, small, isolated D-subnet, such as might be found in the vicinity of a gas or oil well or a very small machine shop, to integrated systems of many thousands of devices and multiple D-subnets that can cover a multi-square-kilometer plant. There is no technical limit on the number of devices that can participate in a network that is composed of multiple D-subnets. A D-subnet, a group of DLEs sharing some DL configuration aspects, may contain up to 30 000 DLEs (which is a limitation of the D-subnet addressing space). With multiple D-subnets, the number of DLEs (and thus devices) in the network can scale linearly.

The maximum amount of higher-layer or management data that can be conveyed in a single DPDU is limited by the PhL and the amount of required DL overhead. Therefore this standard supports fragmentation within the DL, enabling transmission of a much greater amount of data. In fragmentation, the data is segmented into appropriate-size portions at the originating

DLE, encapsulated in DPDUs and transmitted through the D-subnet, then reassembled at the receiving DLE. One use of this mechanism is to update device firmware.

4.6.3 Extensibility

The protocols defined by this standard have fields and parameter value ranges that are reserved for future use, and version (edition) identifiers in headers that permit identification of the appropriate edition.

NOTE These features are intended to permit future revisions of this standard to offer additional or enhanced functionality without unnecessarily sacrificing backward compatibility, and without the encoding bloat that typically occurs through use of the ASN.1 declared extensibility mechanism.

4.6.4 Simple operation

Upon provisioning, as described in Clause 13, a DLE can automatically join the D-subnet and its superior N-network. Automatic device joining and D-subnet formation enables system configuration with minimal need for personnel who have specialized radio frequency (RF) training and tools.

Additionally, this standard supports the use of fully redundant and self-healing D-routing techniques to minimize D-subnet maintenance. (See 9.1.6 for further information.)

4.6.5 Site-license-exempt operation

This standard uses radios compliant with IEEE 802.15.4, using 2,4 GHz DSSS channels 11..26 as specified in that standard.

NOTE 1 The 2,4 GHz ISM band is available and site-license-exempt in most countries, provided that the equipment has a type-license for such operation that is accepted in that country.

NOTE 2 ISA TR100.00.01 provides additional information on radio operation.

4.6.6 Robustness in the presence of interference, including from other wireless systems

This standard uses time-based channel hopping

- to provide a level of immunity against interference from other RF devices operating in the same band,
- to mitigate multipath interference effects,
- to facilitate coexistence with other RF systems, and
- to meet common regulatory requirements.

In some regulatory regimes, coexistence may be further enhanced through selective channel blacklisting, thereby avoiding otherwise-occupied channels within the band. Selective channel blacklisting can also enhance reliability by avoiding the use of channels with consistently poor performance.

4.6.7 Determinism and contention-free media access

This standard defines a time-division-multiple-access (TDMA) mechanism that allows a device to access the RF medium on a schedule, such that much of the competition for use of the channel has been pre-resolved by the scheduling agent. Time-synchronized communication is based on consecutive timeslots that have configurable durations, usually in the range of 10 ms to 12 ms. Scheduling and DLE operation are greatly simplified when all timeslots have a single, common duration, so WISNs usually are configured to have only one timeslot duration that is used for all timeslots.

NOTE 1 Because timeslots are assigned to logical channels that are then mapped cyclically to physical channels, use of a single common timeslot duration means that avoidance of contention on the logical channels automatically avoids that contention on the physical channels. Timeslots of differing durations lose this scheduling simplification.

A sending DLE is assigned a timeslot and channel unique to that device and the device to which it will communicate. These timeslot durations are configurable on a per-superframe basis. A superframe is a cyclic collection of timeslots. The ability to configure timeslot duration enables

- shorter timeslots to take full advantage of optimized implementations;
- longer timeslots to accommodate
  - extended DPDU wait times,
  - serial acknowledgment from two or more configured devices (e.g., duocast), and
  - CSMA/CA at the start of a timeslot (e.g., to implement listen-before-talk, or for prioritized access to shared timeslots); and
- periods of extended duration for slow-channel-hopping.

NOTE 2 Local regulatory requirements may constrain the maximum duration of such a slow-channel-hopping period.

Support is provided for both dedicated time slots for predictable, regular traffic and shared time slots for bursty traffic such as alarms. Publishing/subscribing, client/server communications, alert reporting and bulk data transfer are also supported.

4.6.8 Self-organizing networking with support for redundancy

Fully redundant and self-healing routing techniques, such as mesh routing (see 9.1.6), support end-to-end network reliability in the face of changing RF and environmental conditions. Special characteristics that allow the network to adapt the frequencies used (e.g., adaptive channel-hopping) along with mesh routing, can automatically mitigate coexistence issues without user intervention.

4.6.9 Internet-protocol-compatible NL

This standard's NL uses header formats that comply with the Internet Engineering Task Force's 6LoWPAN standards, thus facilitating potential use of IPv6-compatible networks as backbone networks in support of this standard. Use of headers that comply with 6LoWPAN does not imply either

- that a backbone network needs to be based on 6LoWPAN or IPv6, or
- that a network based on this standard is open to Internet hacking.

In fact, many networks based on this standard will not be directly connected to the Internet. Others may use the older IPv4 standard, at least during initial years of operation.

NOTE Use of IPv6 enables use of standard networking tools and software, as well as the potential for use of a wide variety of IETF standards in future revisions of or extensions to this standard.

4.6.10 Coexistence with other radio frequency systems

4.6.10.1 Coexistence overview

The system architecture specified by this standard is specifically designed to support coexistence with other

- WISNs (i.e., wireless systems conforming to this standard);
- other communication networks operating at 2,4 GHz that employ varying versions of IEEE 802.11, IEEE 802.15.1 and IEEE 802.15.4; and
- other devices that use the same radio frequency spectrum.

Operating with very short, time-synchronized communications tends to reduce congestion of RF bands and to allow neighboring systems to recover quickly from lost or corrupted PhPDUs.

Due to the reduced dwell time on any one channel when channel-hopping, the impact on other radio systems is reduced and reliability in the face of interference is increased. For example, DPDUs may be resent on other, non-interfered channels. Selective channel blacklisting increases coexistence even further by avoiding those channels that are predetermined to be unusable or too congested.

This standard supports (but does not require) the use of clear channel assessment (CCA) to minimize collisions with non-synchronized systems, and also to provide CSMA/CA functionality within synchronized systems.

NOTE Some regulatory jurisdictions require use of CCA in certain modes of operation. Such required use is provided by this standard when a device is configured for operation in those jurisdictions.

The WISN architecture is designed to support operation in the presence of interference from unintentional radiators, such as microwave ovens, using channel-hopping and an automatic repeat-request (ARQ) protocol. ARQ is a common error control method for data transmission that uses acknowledgments for successful message reception, coupled with delayed retransmission in the case of erroneous reception, to achieve reliable data conveyance.

For additional information on diversity techniques that maximize coexistence, see 9.1.2.

4.6.10.2 Coexistence strategies

4.6.10.2.1 General

The following are examples of coexistence techniques that are not specific to any protocol. They improve coexistence with a wide range of devices sharing the 2,4 GHz band while optimizing the success of each communication attempt.

NOTE See IEC 62657-2 for a more comprehensive discussion of wireless coexistence.

4.6.10.2.2 Leverage infrastructure for high data rate communication links

Multi-hop networks convey the same higher-layer data multiple times, once (or more) per hop. One basic capability of this standard is the ability to get the data to a DLE connected to a backbone subnet (preferably one offering a high data rate and low error rate) as directly as possible. This often reduces the use of the PhL specified by this standard to one or two D-transactions and D-subnet hops per conveyed higher-layer PDU.

4.6.10.2.3 Time-slotted operation

Time-slotted operation and scheduled transmissions serve to minimize collisions within the D-subnet, thus avoiding unnecessary use of the channel for retries.

4.6.10.2.4 Radio type selection

The 2,4 GHz subset of IEEE 802.15.4 was selected as the PhL for this standard because, under many conditions, overlapping similar radios, as well as IEEE 802.11 radios, can be active simultaneously without loss of conveyed data.

NOTE This standard is focused primarily on coexistence with the most recent versions of those standards, as older versions tend to be encountered less frequently as the years progress.

4.6.10.2.5 Low-duty cycle

Data conveyance for the focus applications described in 0.1 is infrequent, while added overhead from the conveying protocol layers is minimized.

4.6.10.2.6 Staccato transmissions

Expected transmissions are very short, which is a feature of the selected PhL. This enables co-located IEEE 802.11networks to recover quickly in the event of interference from the WISN.

4.6.10.2.7 Time diversity

Many of the focus applications have less stringent latency requirements than other users of the spectrum, providing more opportunity to use time diversity for coexistence. Configurable retry periods, potentially spanning hundreds of milliseconds, enable the system to coexist with other users that may require use of the same spectrum during higher-priority bursts of activity.

4.6.10.2.8 Channel diversity

The low-duty cycle of the radio is spread across up to sixteen IEEE 802.15.4:2011 channels, further reducing the worst-case potential for interference to 1 % of the time or less under many realistic scenarios.

4.6.10.2.9 Spectrum management

The user may configure superframes within the D-subnet to limit operation to certain radio channels.

4.6.10.2.10 Selective channel utilization

Where the regulatory regime permits, D-management can avoid problematic channels on a link-by-link basis, such as channels exhibiting IEEE 802.11 cross-interference or persistent multipath fades.

4.6.10.2.11 Collision avoidance

All DLEs support CSMA/CA, which allows a DLE to implement a “listen before talk” protocol to provide real-time detection of ongoing use of the channel and delay its own transmission, reducing interference to those other users.

NOTE Such avoidance is required in some regulatory regimes, at least in some modes of operation.

4.6.10.2.12 Varying PhPDUs

Due to the DL’s built-in security measures, which include defense against same-channel and cross-channel replay attacks, PhPDUs vary from transmission to transmission even when retransmitting the same nominal DPDU information. With spread-spectrum modulation this results in time-varying interference even when otherwise-identical messages are being transmitted.

4.6.11 Time-slotted assigned-channel D-transactions as the basis for communication

4.6.11.1 Overview

Except during the interval when a DLE is soliciting the opportunity to join a D-subnet, each instance of DLE communication in accordance with this standard occurs

a) within a prespecified time window, known as a timeslot, relative to the DLE’s sense of TAI time;

b) on a specific PhL channel at a power level that meets local regulations;

c) using a specific timeslot template for the initiator of a D-transaction, which specifies

   1) channel acquisition, configured in accord with local regulations and the timeslot template;

2) transmission of a Data DPDU (i.e., the initial DPDU of a transaction) containing either higher-layer data or management data to a set of intended correspondents; and

3) when so specified by the timeslot template, attempted reception of one or more ACK/NAK DPDUs (i.e., short control signaling) sent by intended correspondents;

NOTE 1 Intentional reception of more than one ACK/NAK DPDU is useful for assessing network operation but is not essential for successful DPDU conveyance.

d) using a different specific timeslot template for an intended correspondent of a D-transaction, which specifies

1) the duration of the channel acquisition phase, related to c)1);

2) attempted reception of a Data DPDU containing either higher-layer data or management data addressed to either the DLE itself or to a specified other DL16Address; and

NOTE 2 This latter capability is used for multicast/broadcast and duocast/N-cast.

3) when reception d)2) did occur and was error-free at the PhL with an error-free DLE FCS, and when so specified by the timeslot template, transmission of a single ACK/NAK DPDU (i.e., short control signaling) sent to the sending DL16Address of the DPDU received in d)2), occurring either

i) at a specified delay after the end of receipt of the Data DPDU d)2), or

ii) at a specified time before the scheduled end of the timeslot,

as specified by the timeslot template.

When the corresponding timeslot template for the transaction initiator specifies more than one interval for ACK/NAK DPDU reception in c)3), then the timeslot templates for the transaction's responders differ in their assigned values for d)3)i) or d)3)ii), thus allocating those potential responses to disjoint time intervals within the timeslot.

The devices to which c)2) applies are known as *transaction initiators*; those devices to which d)2) applies are known as *transaction recipients* (which are the intended recipients, and not just eavesdroppers); those devices to which d)3) also applies are known preferentially as *transaction responders* (although they are also *transaction recipients*).

NOTE 3 IEEE 802.15.4e specifies mechanisms that are similar to, but not identical to, many of the DL mechanisms specified in this standard.

4.6.11.2 Channel acquisition phase

The channel acquisition phase of a transaction has two uses.

a) **ListenBeforeTalk (LBT)**: A mode of operation that is required in certain regulatory jurisdictions and optional in others, whose purpose is to reduce interference with other devices transmitting in the same frequency range, whether those devices use similar PhLs (e.g., other IEEE 802.15.4 systems on the same channel) or different PhLs that overlap in their frequency use (e.g., IEEE 802.11). In this mode of operation, intended transaction initiators sample the channel in a specified way (e.g., CCA mode 1) for a specified time interval, according to local regulatory requirements, and terminate the intended use of the timeslot if that sampling implies that the channel is in use by another device.

b) **CSMA/CA**: A means by which multiple DLEs conforming to this standard attempt to claim use of a designated shared-use timeslot. In this mode of operation each competing transaction initiator operates as in a) for a time interval that is uniformly chosen from a distribution of interval durations that increases exponentially with successive failures (up to some predetermined limit), thus providing exponential backoff in cases of congestion for use of the timeslot. When the smallest value of the selected interval is greater than zero, such operation supports prioritized access because DLEs implementing the CSMA/CA mode tend to defer to those that do not.

The two modes a) and b) can be combined to meet both sets of objectives. The timeslot templates used by intended recipients need to account for these initial delays, as in

4.6.11.1, d)1), so that such recipients do not terminate reception prematurely in an attempt to minimize the energy used by their PhL receivers when active.

Due to well-known properties of RF propagation, the above ListenBeforeTalk and CSMA/CA processes are not reliable in their ability to detect either channel use by other devices or the potential that channel use for one transaction will interfere with other, distant, ongoing communications. This issue has many names, often called the “hidden node” problem. Thus deferral of transactions due to a) and/or b) is always pessimistic with respect to projected interference, yet non-deferral is inadequate to avoid interference (which occurs at receivers) caused by concurrent RF emitters that fail to detect each other.

4.6.11.3 Communication phase

The communication phase of a transaction is used for three basic classes of transactions, which are:

a) **Multicast/broadcast**: The timeslot template for the transaction initiator consists of 4.6.11.1, c)1) and c)2), with c)3) omitted, while the timeslot template for the transaction recipients consists of 4.6.11.1, d)1) and d)2), with d)3) omitted. For these transactions there are no transaction responders, since no opportunity for an immediate response of short control signaling is provided in the template.

b) **Unicast**: The timeslot template for the transaction initiator consists of all parts of 4.6.11.1, c), while the template for the recipients consists of all parts of 4.6.11.1, d). For these transactions there is exactly one intended transaction responder, with a single ACK/NAK immediate response opportunity provided in the template.

c) **Duocast/N-cast**: The timeslot template for the transaction initiator consists of all parts of 4.6.11.1, c), while the template for the recipients consists of all parts of 4.6.11.1, d). For these transactions there is a designated number (2 or n, respectively, for duocast and N-cast) of intended transaction responders, each with a different timeslot template that specifies a single response opportunity for an ACK/NAK DPDU, disjoint from all the other response opportunities for the same transaction.

   In this transaction class all, or all but the first, transaction responders sensitize themselves to receiving a DPDU whose destination DL16Address is not the DLE’s own DL16Address. In such cases the timeslot template also specifies the DL16Address to be used for this purpose. This use of a distinct DL16Address applies only to the Data DPDU of the transaction; any ACK/NAK DPDU uses the actual explicit or implied DL16Addresses of the transaction responder and initiatior.

4.6.12 Robust and flexible security

All compliant networks have a security manager to manage and authenticate cryptographic keys in transit. Security primitives defined by IEEE 802.15.4 are used by the DLE and TLE, providing message originator authentication, message integrity, and optional message content privacy.

Device authentication is enabled by the use of symmetric keys and unique device IDs, with an option for use of asymmetric keys during the device provisioning process and some other security-related processes.

During normal operation, received data authenticity and data integrity are verifiable through the use of secret symmetric keys known to both the originator and the receiver(s).

During provisioning, the authenticity of the received device credentials from a new device may be verified by a system manager through the optional use of public keys shared openly by the new device, and a corresponding asymmetric private key kept secret within the new device.

PDUs are protected using the default AES-128 or an alternative, locally-mandated block cipher, using standard cryptographic modes. Secret symmetric keys that are known to communicating entities are used to secure device-to-device communication.

4.6.13 System management

This standard includes functions to manage communication resources on each individual device, as well as system resources that impact end-to-end performance. System management provides for policy-based management of the runtime configuration and also monitors and reports on configuration, performance, fault conditions, and operational status. The system management functions take part in activities such as:

- device joining and leaving the network;
- reporting of faults that occur in the network;
- communication configuration;
- configuration of clock distribution and the setting of system time;
- device monitoring;
- performance monitoring and optimization.

System security management works in conjunction with the system management function and, potentially, external security systems to enable secure system operation.

All management functions are accessible remotely via the gateway.

4.6.14 Application process using standard objects

This standard's application process is represented as a standard object which contains one or more communicating components drawing from a set of standard defined application objects. These objects provide storage for and access the data of an application process.

Defining standard objects provide an open representation of the capabilities of a distributed application in a definitive manner, thereby enabling independent implementations to interoperate. Objects are defined to enable not only interaction among field devices but also interoperation with different host systems.

The standard objects and services of this standard may be used to directly map existing legacy field device communications onto standard objects and application sublayer communication services, thereby providing a means to adapt legacy devices to communicate over the WISN.

4.6.15 Tunneling

The native protocols defined by this standard allow devices to encapsulate foreign PDUs and transport these foreign PDUs through the WISN to a destination device within the WISN, which usually is a gateway to legacy protocols. This encapsulation mechanism is referred to as tunneling. Successful application of tunneling depends upon how well the foreign protocol's technical requirements (e.g., timing, latency, etc.) are met by the instantiation of the WISN.

5 System

5.1 General

In this standard a system is defined to have an application focus and addresses applications and their needs. Networks, on the other hand, have a communication focus and are devoted to the task of device-to-device communication. For the purposes of this standard, a network is a component of a larger system.

Clause 5 describes how the various protocol layers and functions of this standard work together to form a system that achieves the goals of this standard. Specifically, Clause 5 describes the system aspects of devices, networks, protocol suite, data flow, a shared time base, and the applications need for firmware revisions.

5.2 Devices

5.2.1 General

Devices implement a combination of protocol layers, usually including a PhLE, a DLE, a NLE, a TLE and an ALE, and may include functions such as the system manager role, the security manager role, one or more gateway roles, and support for provisioning other devices in the network.

NOTE Only system behaviors are specified in Clause 5.

5.2.2 Device interworkability

Device interworkability is the ability of devices from multiple vendors to communicate and maintain the complete network. Device interworkability requires control over the device's various options, configuration settings, and capabilities.

a) **Options**: To allow all devices to interwork, and to interoperate within a constrained domain of application, regardless of implemented options (those defined within this standard), devices shall be capable of disabling (i.e., not using) any options that are not mandatory for the device's configured role(s), as specified in the role profiles of Annex B.

b) **Configuration settings**: The system manager is responsible for configuring WISN devices and roles implemented by WISN devices. The system manager is described in Clause 6. Some configuration aspects are described in Annex D.

c) **Capabilities**: There are minimum capabilities to be met for devices based upon their role in the system. Annex B defines the baseline capabilities required for all devices.

5.2.3 Profiles

A profile can be described as a vertical slice through the protocol layers. It defines those options in each protocol layer that are mandatory for that profile. It also defines configurations and parameter ranges for each protocol. The profile concept is used to reduce the risk of device interworkability and interoperability problems between different manufacturers' products. Interoperability in areas outside the scope of this standard requires either use of profiles beyond those of this standard, or other extra-standard arrangements.

A role profile is defined as the baseline capabilities, including any optional features, settings, and configurations, that are required of a device to perform that role adequately. The roles are defined in 5.2.6.2.

5.2.4 Quality of service

An application within a device is assumed to know the level of service that is necessary for its proper operation. The level of quality of service (QoS) is agreed upon via a contract between the system manager and the requesting device. When the application within a device desires to communicate at a certain QoS level, it sends a request to the system manager notifying it that it wishes to communicate with a specific destination and that it desires a given QoS level. This desired QoS level is indicated by a desired contract and message priority. In addition, a certain level of reliability, periodicity, phase, and deadline for periodic messages, and short-term burst rate, long-term burst rate, and maximum number of outstanding requests for client/server messages, may be indicated. See 6.3.11.2.7 for specific information on QoS.

5.2.5 Device worldwide applicability

5.2.5.1 General

This standard is designed to support operation within a fixed geographic area that operates under uniform regulations. As such it is intended to support operation in the 2,4 GHz band anywhere in the world, as discussed in 9.1.15.6. For example, Annex V discusses how systems and devices can be configured to meet EU regulatory constraints.

This standard also is designed to support wireless automation systems operating on mobile platforms, such as marine vessels (e.g., container ships and petrochemical tanker ships) and trains that can move between geographic regions (e.g., countries) where differing, and perhaps conflicting, regulations apply. For example, a container ship usually would be subject to local regulations when in port, and thus could have different compliance requirements when in Rotterdam than when in Tokyo Bay, because the regulations that apply to wireless systems when operating under EU jurisdiction differ from those that apply when operating under Japanese jurisdiction.

Radio regulations often require devices to operate at constrained power levels at all times, including during over-the-air provisioning. Some identified EIRP and EIRP density limits are: 10 mW/MHz (Japan); 10 dBm (China); 10 dBm, 10 mW/MHz and 20 dBm (EU); 36 dBm with at most a 6 dBi antenna gain (US FCC).

In some countries, such as France, emission levels on certain channels may need to be attenuated. In other countries, such as Korea, the number and range of channels needs to be constrained.

5.2.5.2 Operation within a fixed regulatory regime

The dlmo.CountryCode field, described in 9.1.15.6, is used to specify the regulatory regime. It may also be used to specify some overriding regulatory constraints.

This field includes a “self-locking” mechanism, so that it is possible to set the value of this field to make the entire field unchangeable while the device is operational. Once set, only reprovisioning the device, such as after a repair or transfer of ownership, is able to clear that lock.

This “set and forget” feature was included to support regulatory regimes, such as some within the EU, where no operational method may override the RF emission limits established under regulation. However, the feature is provided in a way that also supports device repair and resale into other regulatory jurisdictions, whose requirements might differ from those of the jurisdiction into which the device was previously deployed.

5.2.5.3 Operation on a platform that moves between regulatory regimes

Some wireless automation systems may be located on a mobile platform such as a container ship or a petrochemical tanker ship or a consist of railroad tank cars that moves between regulatory regimes, operating temporarily in each. That transition between regimes can occur rapidly, as when a train crosses a border, or slowly, as when a ship transits from national waters to international waters.

This standard is designed to support operation of wireless systems in such mobile transport situations, by providing a means by which a single equipment parameter can be changed in each device, for example by a timed action downloaded in advance to each device, to cause all affected devices to change regulatory regimes, after which their operations are constrained by the regulations for the newly-entered regulatory regime.

NOTE Such a change in wireless emission characteristics often will be accompanied by a change in link schedules, for example, to use more or fewer routers in a multi-hop path, or to have available more spare timeslots for retry of transactions that were aborted due to LBT-detected activity in the channel, to better match system operation to the more strict (or relaxed) requirements of the new regulatory regime.

5.2.6 Device description

5.2.6.1 General

Within this standard, devices are the physical embodiment of the behaviors, configuration settings, and capabilities that are necessary to implement and operate a network. There are many different types of devices depending upon the application, environment, and function within the network. To fully describe necessary network behavior without defining specific

device implementations this standard defines roles, protocol layers, and a field medium that devices may embody.

A role defines a collection of functions and capabilities. This standard defines all the roles necessary for the network to operate properly, including system manager, security manager, gateway, backbone router, system time source, provisioning, router, and I/O device. All devices conforming to this standard shall implement at least one role; however, a device may implement many roles. A device implementing a role shall implement all functions required for that role in 5.3.

The protocol layers describe required behaviors. Not all devices are required to implement all the protocol layers defined in this standard. However, all devices conforming to this standard shall implement the network and transport layers in addition to the DMAP functionality as described in 6.2. Every device shall contain a device management function and a device security management function that cooperate with the system processes to enable secure management of a device's resources and the device's usage of system resources.

A field medium is represented within a device by a combination of a PhLE and a DLE, both as described in this standard. While not all devices need to implement a field medium, any device that implements the I/O, routing, or backbone routing roles shall directly support at least one field medium as specified by this standard.

Figure 4 illustrates the distinction between physical devices (e.g., as supplied by a manufacturer) and the roles that those devices can assume.

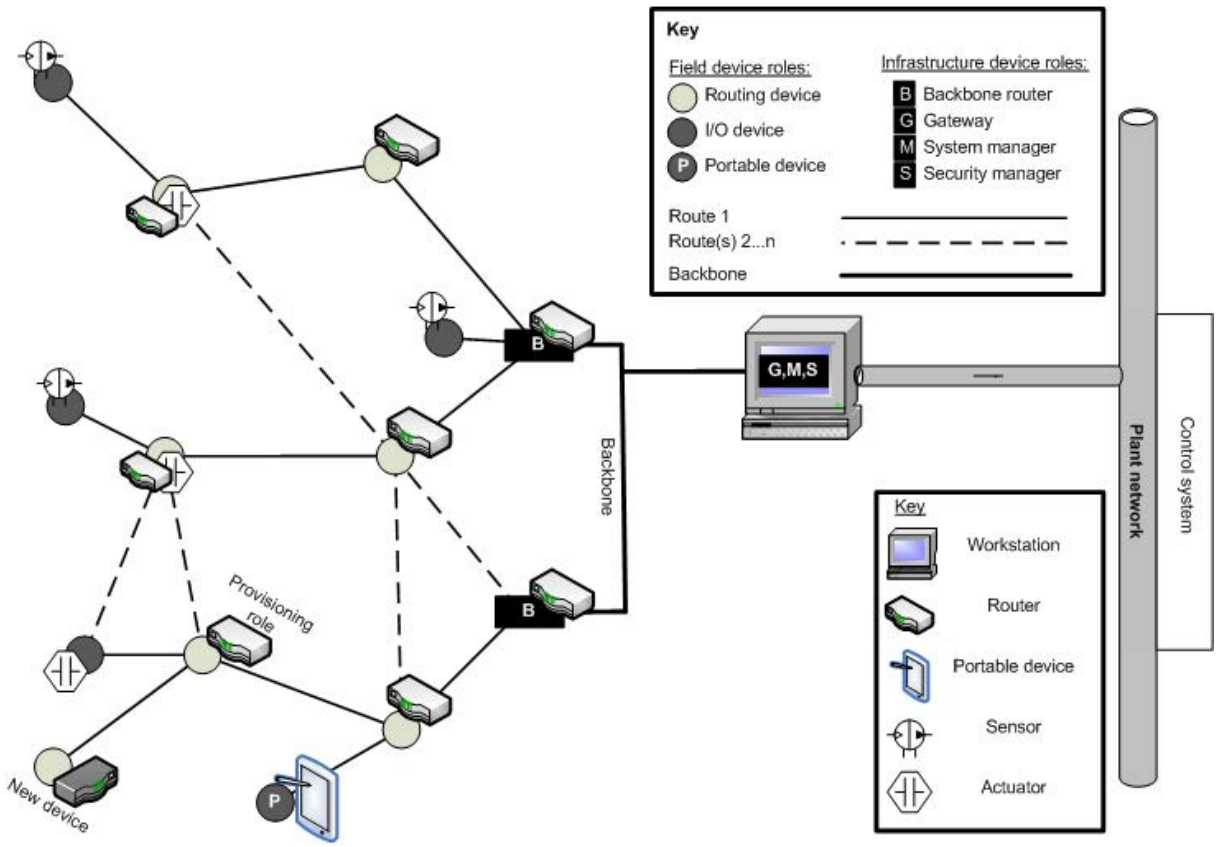


Figure 4 – Physical devices versus roles

Figure 4 shows a representative yet complete network compliant with this standard. Within this network are several types of devices, including sensors, actuators, routers, a handheld computer, and a workstation. As shown in Figure 4, each of these devices may assume different roles within the network. In this example

- the workstation has assumed the roles of gateway, system manager, and security manager, where those roles are described in 5.2.6.10, 5.2.6.11, and 5.2.6.12, respectively;
- two devices have assumed the role of backbone routers (described in 5.2.6.9), while seven other devices have assumed the role of routers (described in 5.2.6.7);
- three sensors, one actuator and a portable computer have assumed the singular role of an I/O device (5.2.6.6);
- the router at the lower left of Figure 4 has assumed a provisioning role, as described in 5.2.6.8, and will provision the new device being introduced; and
- two actuator devices have assumed both the router and I/O roles.

NOTE 1 Although Figure 4 shows the use of a backbone network, the functionality of the backbone network is not specified within this standard.

NOTE 2 The physical devices and roles shown in Figure 4 are intended only as examples.

5.2.6.2 Field medium

5.2.6.3 General

This standard defines one specific field medium, Type A. A field medium type defines the protocol for the PhL and lower DL (i.e, the MAC). Future revisions of this standard may support multiple field media types.

5.2.6.4 Type A

The Type A field medium consists of the PhL and DL as specified by Clause 8 and Clause 9 of this standard.

Devices implementing the Type A field medium and the DL shall implement configuration settings for Radio Silence. The Radio Silence configuration is used to restrain the radio from transmitting during inappropriate times, such as when transmission is unsafe or when regulations prohibit radio transmissions. The Radio Silence configuration settings are defined in 9.1.15.4.

5.2.6.5 Role definitions

5.2.6.6 Input/output

A device with the I/O role shall provide (source) data to or use (consume) data from other devices (and may both provide and use data) and shall have at least one user application process (UAP) object. A device with only an I/O role is a device that has the minimum characteristics required to participate in a network compliant with this standard. The I/O role provides no mechanism for the forwarding of messages or routing for any other device. This enables the construction of devices with the least complexity and the potential for low energy consumption, since they need not expend energy routing other devices' messages, nor are they required to accept and provision new devices wishing to join the network.

NOTE A data source supplies data. An actuator would be an example of a consumer of data (i.e. sink), whereas a sensor would supply data (i.e. source).

Devices that implement the I/O role shall implement the Type A field medium.

5.2.6.7 Router

A device with the router role shall have routing capability, shall act as a proxy, and shall have clock propagation capability. These devices can provide range extension for a network and path redundancy and may provide different levels of QoS on a message-by-message basis. The system manager may disable the routing capabilities of the router role to optimize system performance requirements such as message latency or battery consumption.

Devices that implement the router role shall implement the Type A field medium.

5.2.6.8 Provisioning

A device with the provisioning role (provisioning device) shall be able to provision a device set to factory defaults and shall implement the device provisioning object (DPO; see Clause 13). The provisioning device inserts the required configuration data into a device to allow a device to join a specific network. Devices implementing the PhL shall be capable of being provisioned using the defined physical interface. This capability can be disabled (see Clause 13).

Devices that implement the provisioning role shall implement the Type A field medium.

5.2.6.9 Backbone router

A device with the backbone router role shall have routing capability via the backbone, and shall act as a proxy using the backbone. Backbone routers enable external networks to carry native protocol by encapsulating the PDUs for transport. This allows a network described by this standard to use other networks, including longer-range or higher-performance networks.

While the media and protocol suites of backbone networks are not defined in this standard, it is believed that many instantiations of the backbone router will be with internet protocol (IP) networks. Many of these backbone networks may conform to IPv4 as opposed to the newer IPv6. Clause 10 describes how a WISN NPDU received by a BBR at the BBR's WISN DLE interface is converted into a fully compliant IPv6 NPDU. If the BBR's backbone interface implements IPv6, then the NPDU may simply be routed using standard IPv6. If the BBR's backbone interface implements IPv4, then the BBR shall support the use of IETF RFC 2529 to route the NPDU across the IPv4 backbone.

Devices implementing the backbone router role shall implement the Type A field medium in addition to the BBR's backbone network interface.

5.2.6.10 Gateway

A device with the gateway role implements a high-side interface. An example of an internal GIAP supporting such a high-side interface is given in Annex U. The gateway communicates over the WISN by native access and/or tunneling. Such a device shall have a UAP. The gateway role provides an interface between the WISN and the plant network, or directly to an end application on a plant network. More generally, a gateway marks the transition between communications compliant with this standard and other communications and acts as a protocol translator between an AE described by this standard and other AEs. There can be multiple gateways in a system.

5.2.6.11 System manager

A device implementing the system manager role shall implement the SMAP (6.3.2) and shall set the time source tree.

The system manager is a specialized function that governs the network, devices, and communications. The system manager performs policy-based control of the network runtime configuration, monitors and reports on communication configuration, performance, and operational status, and provides time-related services.

When two devices need to communicate, they do so using a contract. A contract is an agreement between the system manager and a device in the network that involves the allocation of network resources by the system manager to support a particular communication need of this device. This contract is made between the applications in both devices and the system manager. The system manager will assign a contract ID to the contract, and the application within the device will use the contract for communications. An application may

only request the creation, modification, or termination to a contract. It is the sole responsibility of the system manager to create, maintain, modify, and terminate the contract.

For more information on system management, see Clause 6.

5.2.6.12 Security manager

The system security management function, or security manager, is a specialized function that works in conjunction with the system manager and, potentially, external security systems to enable secure system operation. The security manager is logically separable from the system manager, and in some use cases will be resident on a separate device and in a separate location. Every system compliant with this standard shall have a security manager. For more information on the security manager, see Clause 7.

NOTE The communication protocol used between the system manager and the security manager is not defined by this standard because such components are usually supplied by a vendor as a matched pair/set.

For more information on the security management functionality, see 7.7.

5.2.6.13 System time source

A device implementing the system time source role shall implement the master time source for the system. A sense of time is an important aspect of this standard; it is used to manage the device operation. The system time source provides a sense of time for the entire system. This is described in more detail in 6.3.10.1.

Devices implementing the system time source role shall implement any of the I/O, router, backbone router, system manager, or gateway roles.

5.2.7 Device addressing

Each device that implements the Type A field medium shall be assigned a DL16Address for the D-subnet, which is used for local addressing. Each device shall have an EUI64Address that is unique. See Clause 9 for further information.

Each device shall also have an IPv6Address that is assigned by the system manager as described in 6.3.5. The system manager may choose to assign the IPv6Address as a logical address to maintain ALE linkage in the event of a device replacement. The IPv6Address may be used by the application to reach a particular device within a system after the subnet joining process is complete. See Clause 10 for further information.

5.2.8 Device phases

5.2.8.1 General

A device may go through several phases during its operational lifetime. Within each of these phases are multiple states. A notional representation of the phases of the life of a device is shown in Figure 5. See Figure 136 and Figure 137 for normative detail.

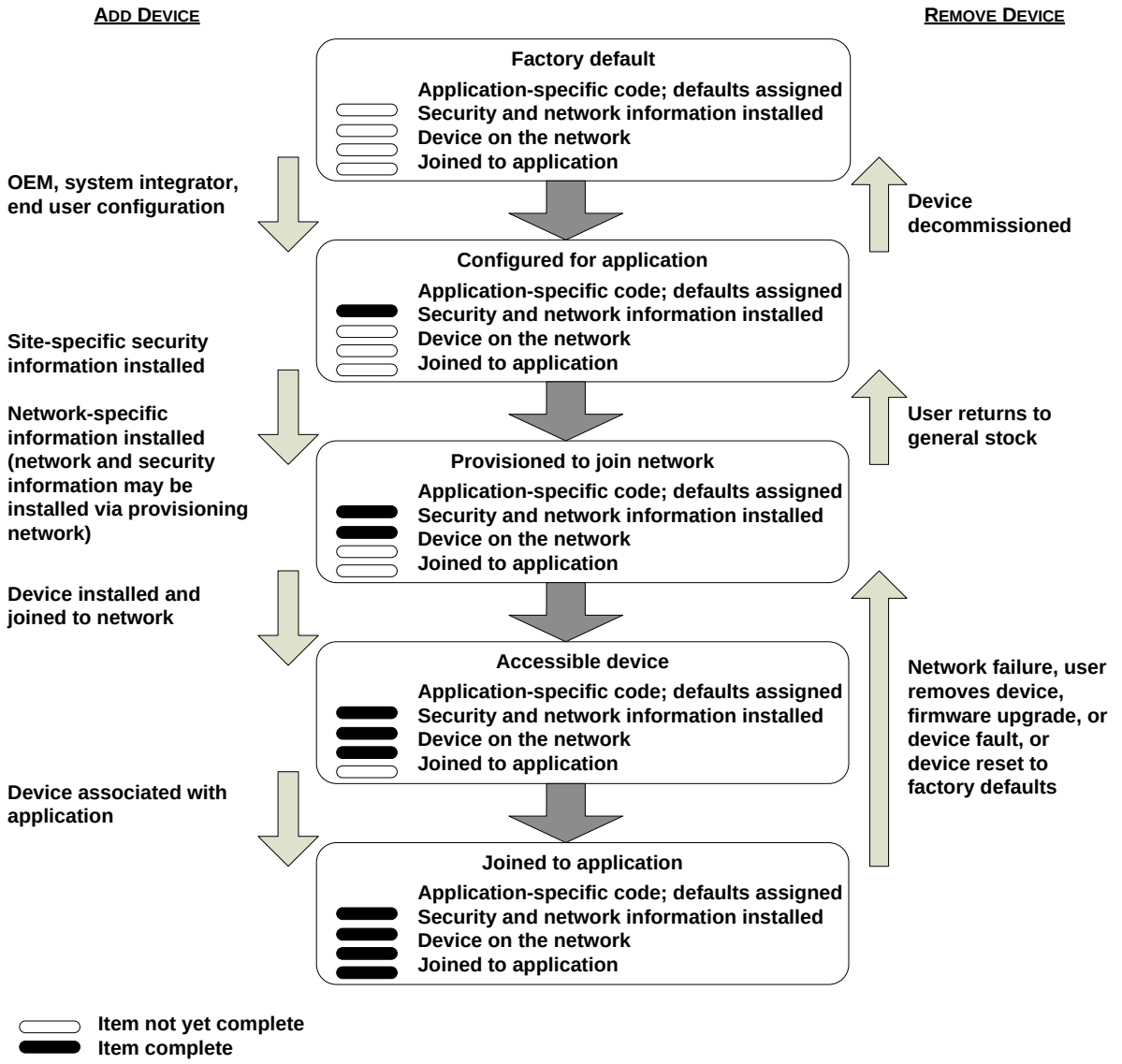


Figure 5 – Notional representation of device phases

A device can pass through these phases several times as it is commissioned and used, then decommissioned and re-commissioned for a different application. After joining the network, devices shall be able to report their status so that applications know whether a device is accessible and whether it is joined to an application.

5.2.8.2 Factory default

A device is considered non-configured if it has not been configured or commissioned with any application- or network-specific information. A non-configured device may come from a manufacturer or may enter a non-configured state as a result of decommissioning.

5.2.8.3 Configured for application

A device is considered configured for an application when it has received its own application-specific programming and when all appropriate defaults have been applied. A device configured for application may come from a manufacturer or may be supplied by a systems integrator or other value added reseller, already provisioned for the intended application. Over-the-air application program updates can occur, but are handled by the device ALE.

5.2.8.4 Provisioned to join the network

A device is provisioned to join the network when it has obtained the appropriate security credentials and network-specific information. A device will usually enter this phase when it has been prepared for installation in an automation application. Often the device will not communicate directly with the security manager; instead, the system manager serves as a relay for all communication with the security manager.

5.2.8.5 Accessible device

A device is considered accessible when it has joined the network and has been authenticated by the system manager. An accessible device can communicate with the system manager.

5.2.8.6 Joined to application

In this phase, an application object on the device can send or receive information to or from the desired application objects on peer devices. See 7.4 for additional detail on the subnet joining process.

NOTE Application objects in any two devices on the network are able to communicate with one another. Refer to Clause 12 for more detail.

5.2.9 Device energy sources

This standard does not restrict the types of energy sources a device may use. The standard allows for energy efficient device behavior that accommodates device operation for long periods of time (e.g., five to ten years) using suitable batteries.

The types of energy sources may be grouped into five categories:

- mains;
- limited battery (e.g., button cell);
- moderate battery (e.g., lead acid);
- rechargeable battery;
- environmental or energy-scavenging.

Devices implementing the roles of I/O or router may be expected to use any category of energy source. The roles of security manager, system manager, gateway, and backbone router are usually performance intensive; therefore devices implementing these roles are recommended to have more capable energy sources such as the mains or moderate battery categories.

The energy source status of devices is critical to proper system management. All devices shall provide energy supply information to the system manager. This information may be used in making routing decisions. See Clause 9 for further details.

5.3 Networks

5.3.1 General

The focus of networks is device-to-device communication. There are numerous aspects of the networks' ability to communicate. These aspects include the atomic (i.e., minimal or irreducible) network, network topologies, device relationships within a network, protocol suite structure, and the concept of shared time.

5.3.2 Minimal network

A minimal network is a network with the minimum amount of devices implementing the minimum number of roles. Although a minimum system could be constructed with just a

system manager and a security manager, a more practical minimum system would include the roles of system manager, security manager, provisioning, system time source, and I/O. The system manager and security manager are two separate roles and may reside in the same device or may be split between two physical devices. A single physical device may assume multiple roles. Therefore, a minimal network shall consist of two devices communicating with each other, where one device implements the roles of system manager and security manager; the roles of provisioning, system time source, and I/O are implemented by either of the devices.

A small representative network of four field devices and one infrastructure device is shown in Figure 6. Although such a network is atypical, it represents a small compliant system. In this network, a single physical device has assumed the roles of gateway, system manager and security manager.

5.3.3 Basic network topologies supported

5.3.3.1 General

Figures 6 to 9 in 5.3.3 provide several informative examples and illustrate the flexibility of the system architecture. The set of examples is not intended to be exhaustive. These examples are presented here only to provide a better understanding of the system elements.

5.3.3.2 Star topology

This standard supports a simple star topology, as shown in Figure 6.

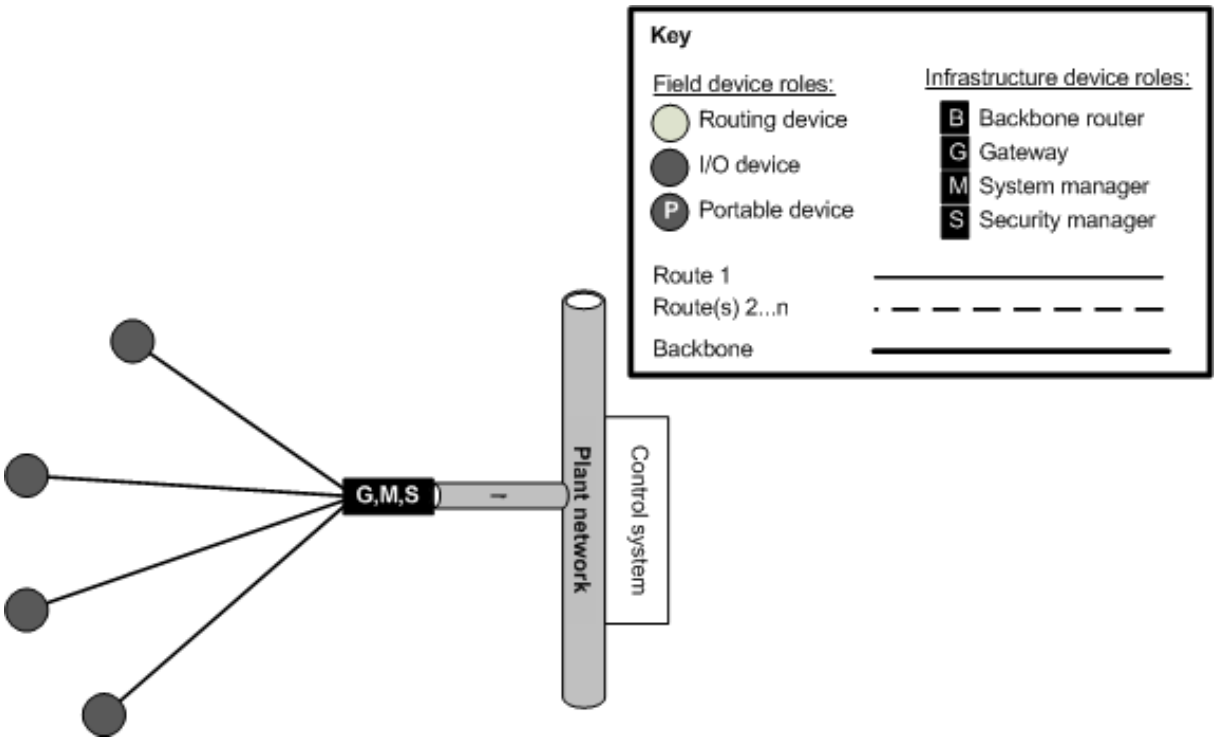


Figure 6 – Simple star topology

This system configuration can yield the lowest possible latency across the physical layer. It is architecturally very simple, but is limited to the range of a single hop.

Within Figures 6 to 9 in 5.3.3, each box labeled G,M,S represents a collection of three separate roles combined into one physical device in the relatively simple networks depicted:

- a gateway;

- a system manager; and
- a security manager.

5.3.3.3 Hub-and-spoke topology

Expanding the network using the backbone routers allows the user to construct a hub-and-spoke network, as shown in Figure 7, wherein devices are clustered around each backbone router, providing access to the high-speed backbone.

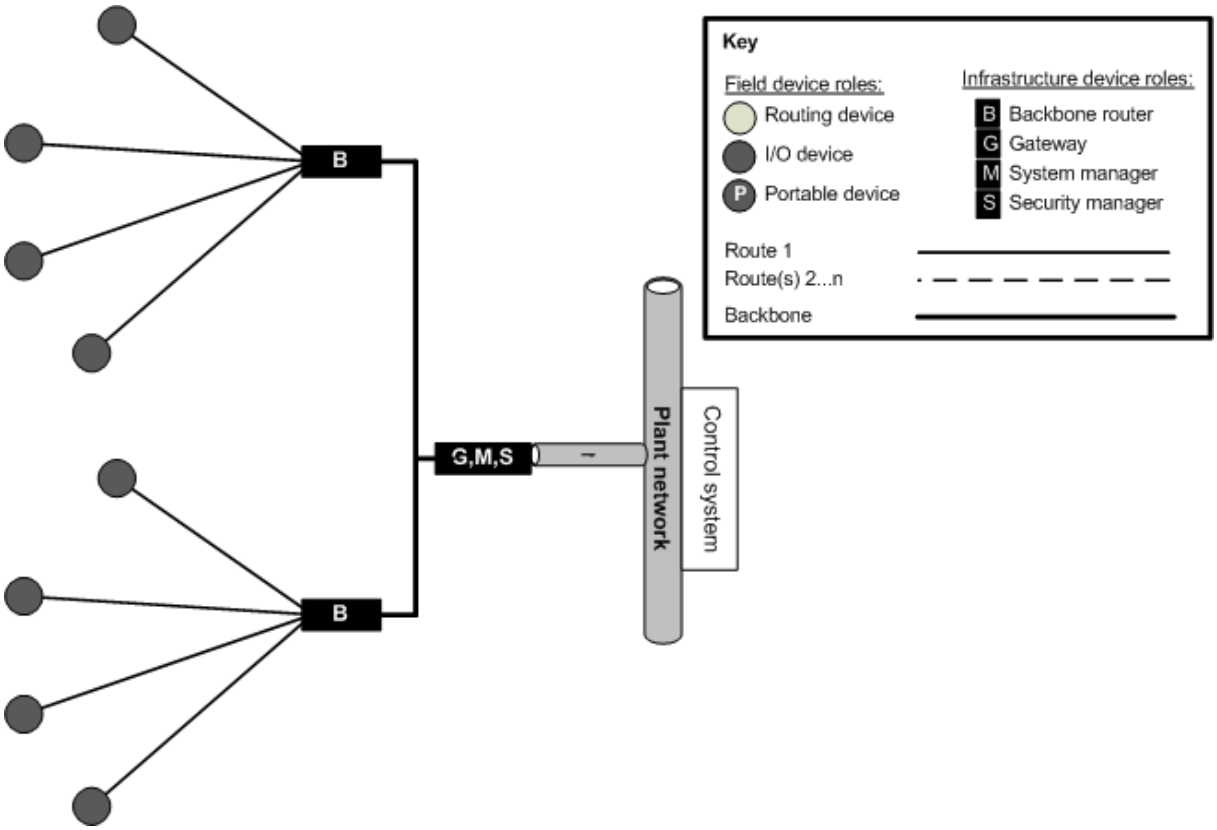


Figure 7 – Simple hub-and-spoke topology

In this case, latency is slightly degraded from the simple star topology, but overall throughput can increase, and in larger systems, average latency can decrease because of the multiple data pipes available (one through each backbone router). Although the network can expand further away from the gateway, it is nonetheless limited to single-hop range around a backbone router.

5.3.3.4 Mesh topology

This standard supports mesh networking topologies, as shown in Figure 8.

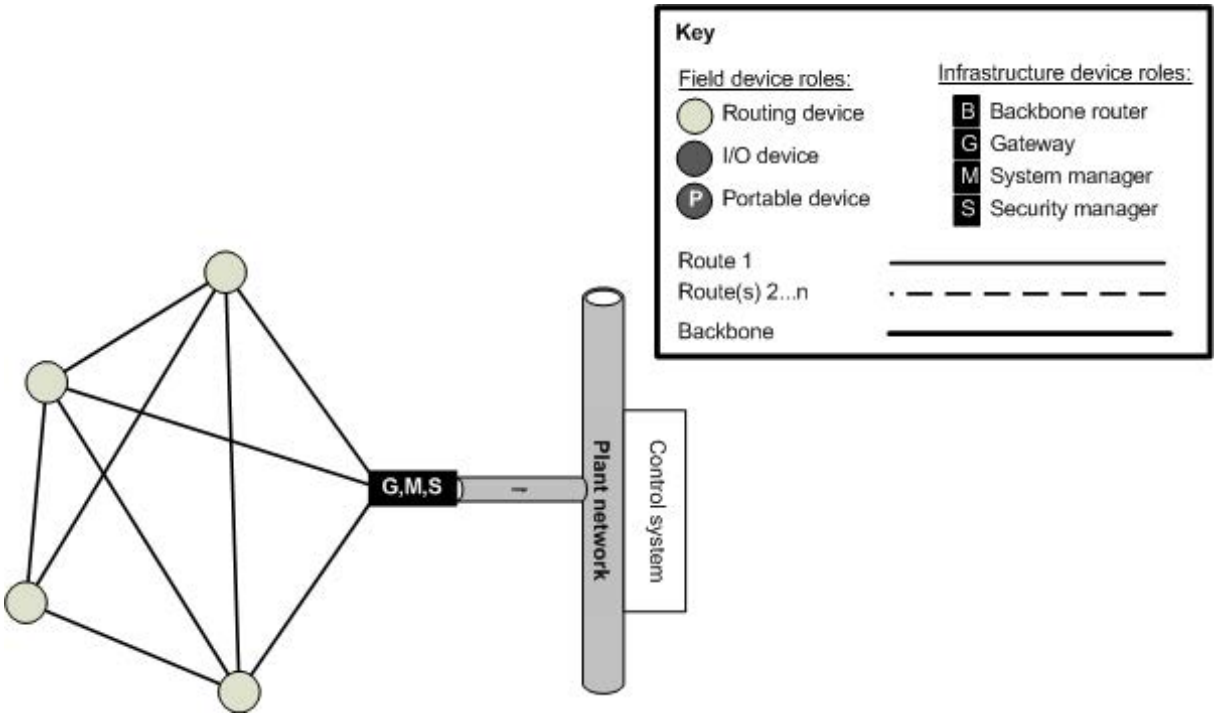


Figure 8 – Mesh topology

In some cases, the number of routes a device can support may be limited. Range is extended as multiple hops are supported. Latency is larger, but can be minimized by proper scheduling of transmissions. Throughput is degraded as device resources are used in repeating messages. Reliability may be improved through the use of path diversity.

For more information on mesh topology, see 9.1.14.

5.3.3.5 Star-mesh topology

The star topology combined with the mesh topology is shown in Figure 9.

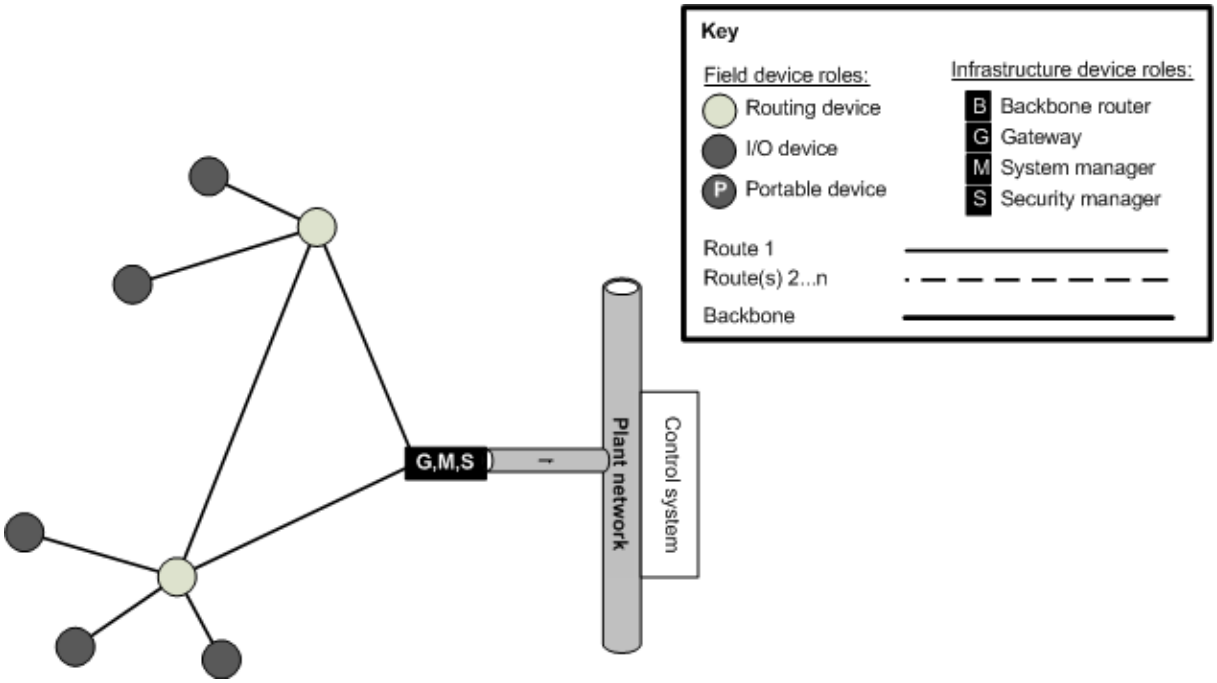


Figure 9 – Simple star-mesh topology

This configuration has the advantage of limiting the number of hops in a network. It does not have the added reliability that full mesh networking can provide.

5.3.3.6 Combinations of topologies

This standard allows for the combination of any of the previously mentioned topologies, so that a configuration can be constructed that best satisfies the needs of the application. For example, monitoring systems that span large physical areas within a plant may use the star-mesh topology or a combination of hub-and-spoke and star-mesh topologies, whereas certain control applications where latency is critical may benefit from a pure star or hub-and-spoke topology. The flexibility of the system allows for all of these topologies to operate in harmony, in any combination.

5.3.4 Network configurations

5.3.4.1 General

The D-subnet in this standard comprises one or more groups of wireless devices, with a shared system manager and (when applicable) a shared backbone. While a D-subnet stops at the backbone router (see 5.5.6), network routing may extend into the backbone and plant network. A complete network includes all related D-subnets, as well as other devices connected via the backbone, such as a gateway, system manager, or security manager. Figure 10 and Figure 11 illustrate the distinction between a D-subnet and a network.

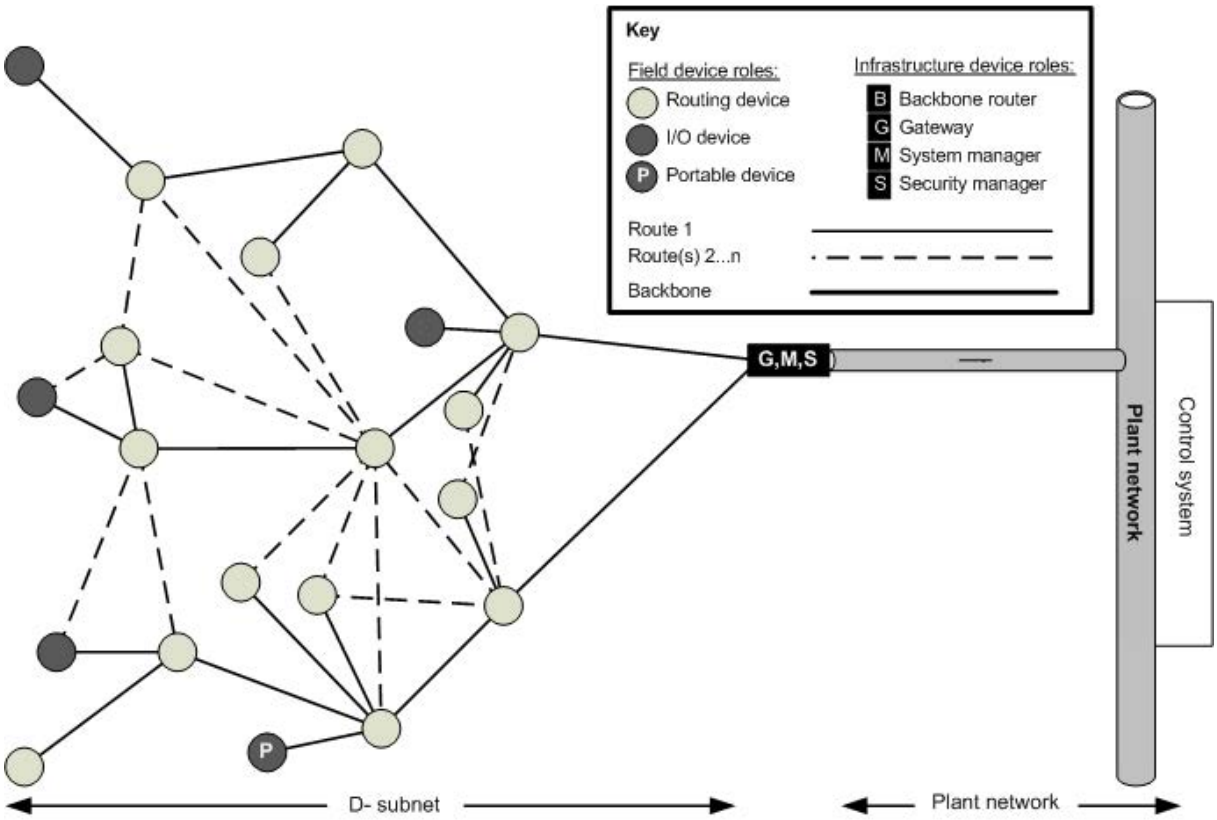


Figure 10 – Example where network and D-subnet overlap

Figure 10 illustrates a simple network comprised of a collection of wireless devices called a D-subnet and additional devices that manage the D-subnet and connect it to other networks. In Figure 10, the network and the D-subnet are the same.

The D-subnet is comprised of both routing and I/O devices. The solid lines between devices designate the first route established between devices, while dotted lines designate the second route, the third route, and so on. Messages may be routed using any one of the known routes.

In Figure 11, the D-subnet includes a collection of field devices up to the backbone routers (boxes labeled B). Backbone routers use connections to a network backbone to reduce the number of hops that messages would otherwise require; this can improve reliability, reduce latency, and extend the coverage of the network.

The network in Figure 11 includes the D-subnet, as well as the backbone and a gateway, system manager, and security manager, which are co-located on the backbone.

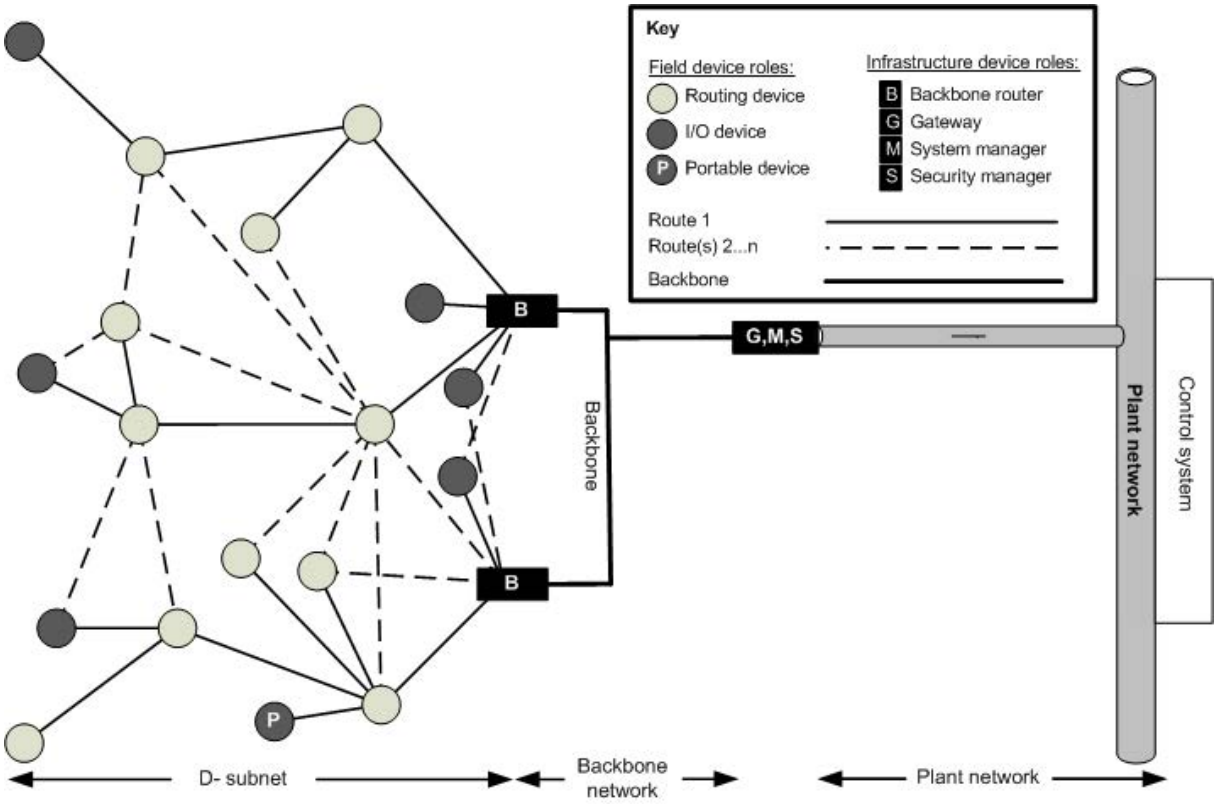


Figure 11 – Example where network and D-subnet differ

5.3.4.2 Multiple gateways – Redundancy and additional functions

Figure 12 illustrates a different physical configuration with three gateway devices. One of the gateway devices also implements the system manager and security manager functions.

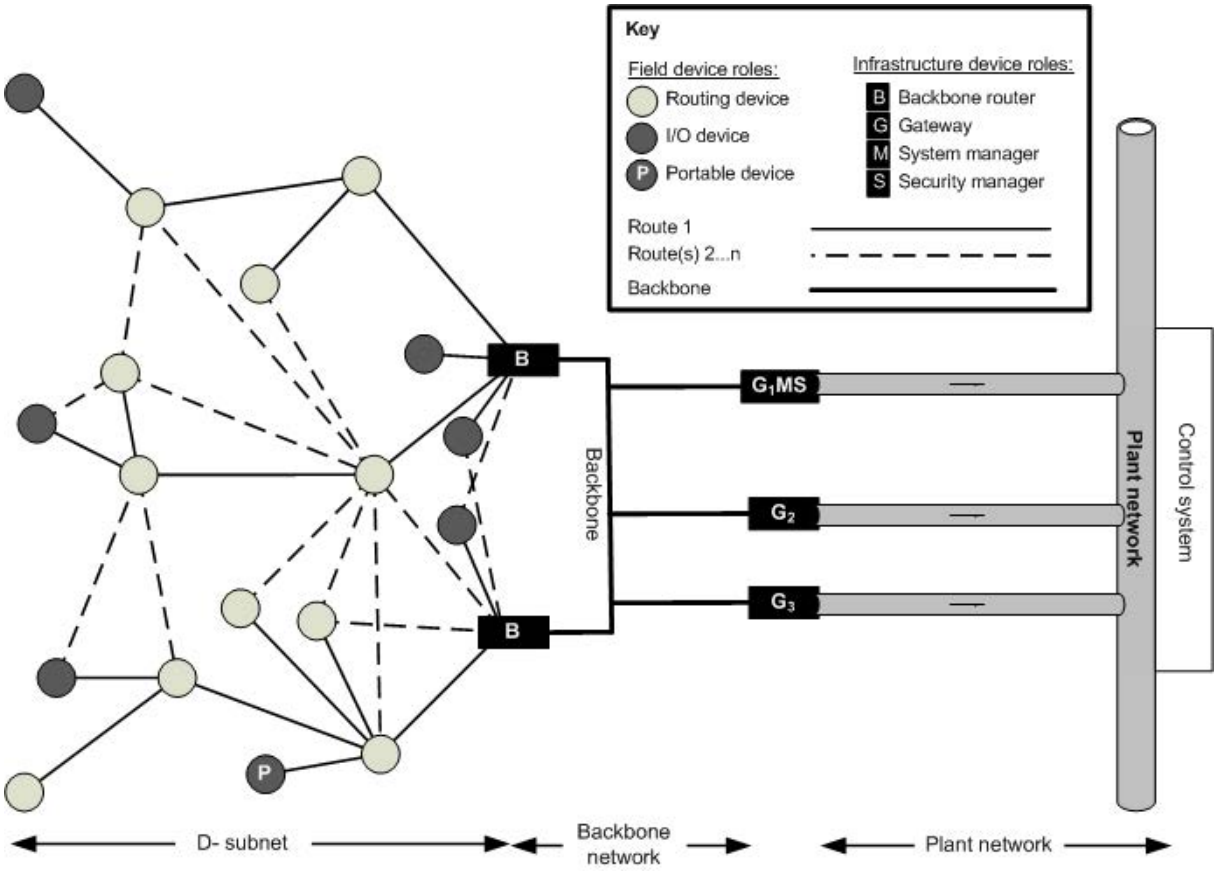


Figure 12 – Network with multiple gateways

The gateway devices may be identical (i.e., mirrored, for redundancy) or unique, for example, with each gateway implementing a software application to handle communications between a particular class of device and a control system attached to the plant network.

5.3.4.3 Multiple gateways – Designating a gateway as a backup

NOTE This standard does not define the functionality of a backup gateway nor the mechanisms for synchronization of backup gateways.

Figure 13 is similar to Figure 10, but with a second G,M,S device (gateway, system manager, and security manager).

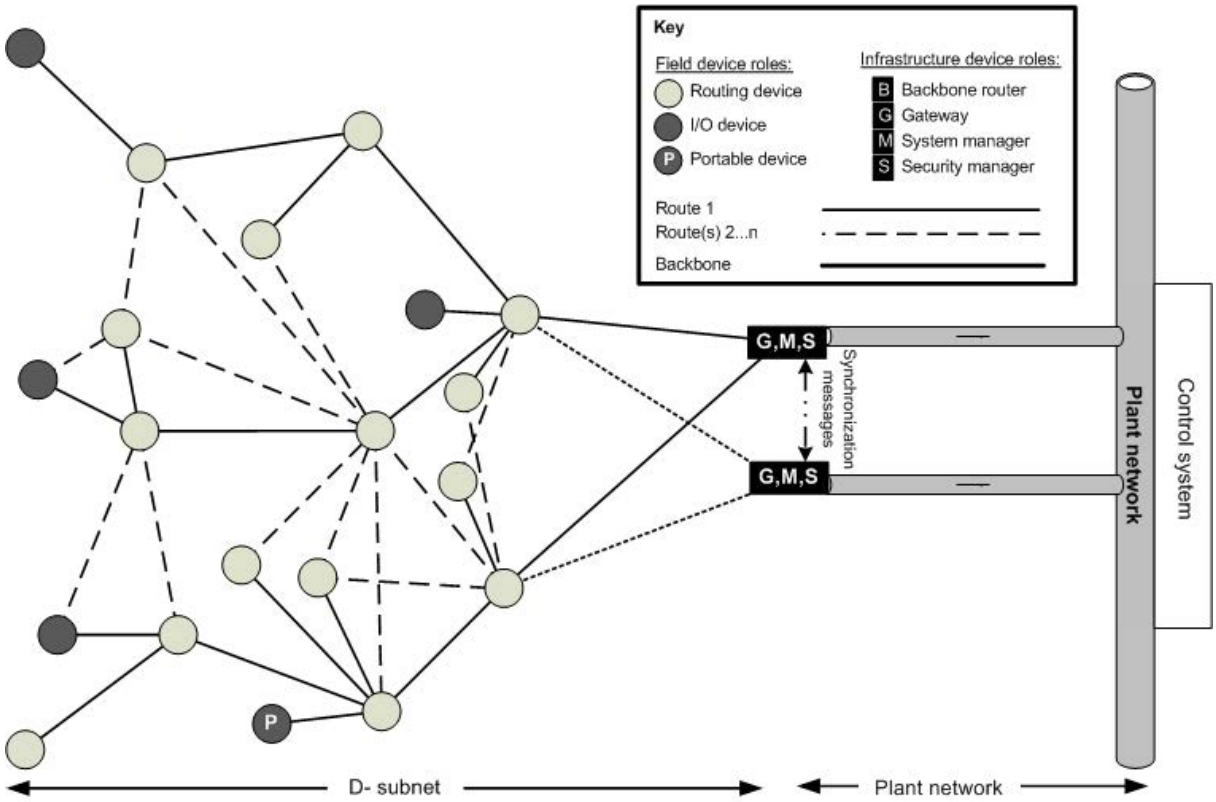


Figure 13 – Basic network with backup gateway

The two G,M,S devices offer identical functionality and may coordinate their operation via synchronization messages exchanged through a backchannel mechanism not specified by this standard. A single G,M,S device may be responsible for all gateway, system manager, and security manager functions, with a second G,M,S device acting as an active standby that remains idle until it is needed. Alternatively, the two G,M,S devices may divide the workload between them until one fails.

5.3.4.4 Adding backbone routers

To the basic network shown in Figure 13, Figure 14 adds backbone routers (boxes labeled B), which facilitate expansion of networks compliant with this standard, in terms of both the number of devices and the area the network occupies.

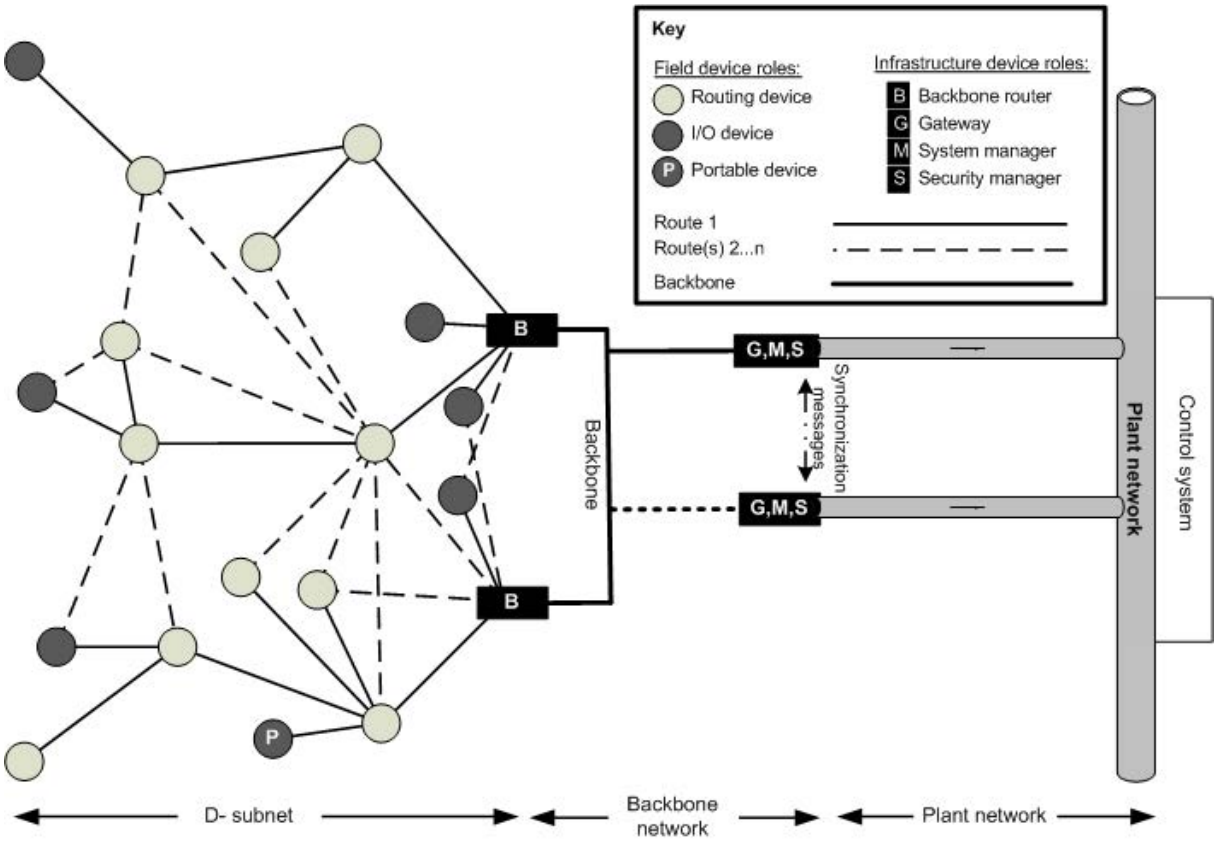


Figure 14 – Network with backbone

5.3.5 Gateway, system manager, and security manager

As shown in Figure 15, the functional roles fulfilled by the G,M,S device in Figure 10, Figure 11, Figure 12, and Figure 14 may be split into multiple physically separated devices, so that the gateway G, system manager M, and security manager S each operate on a separate device.

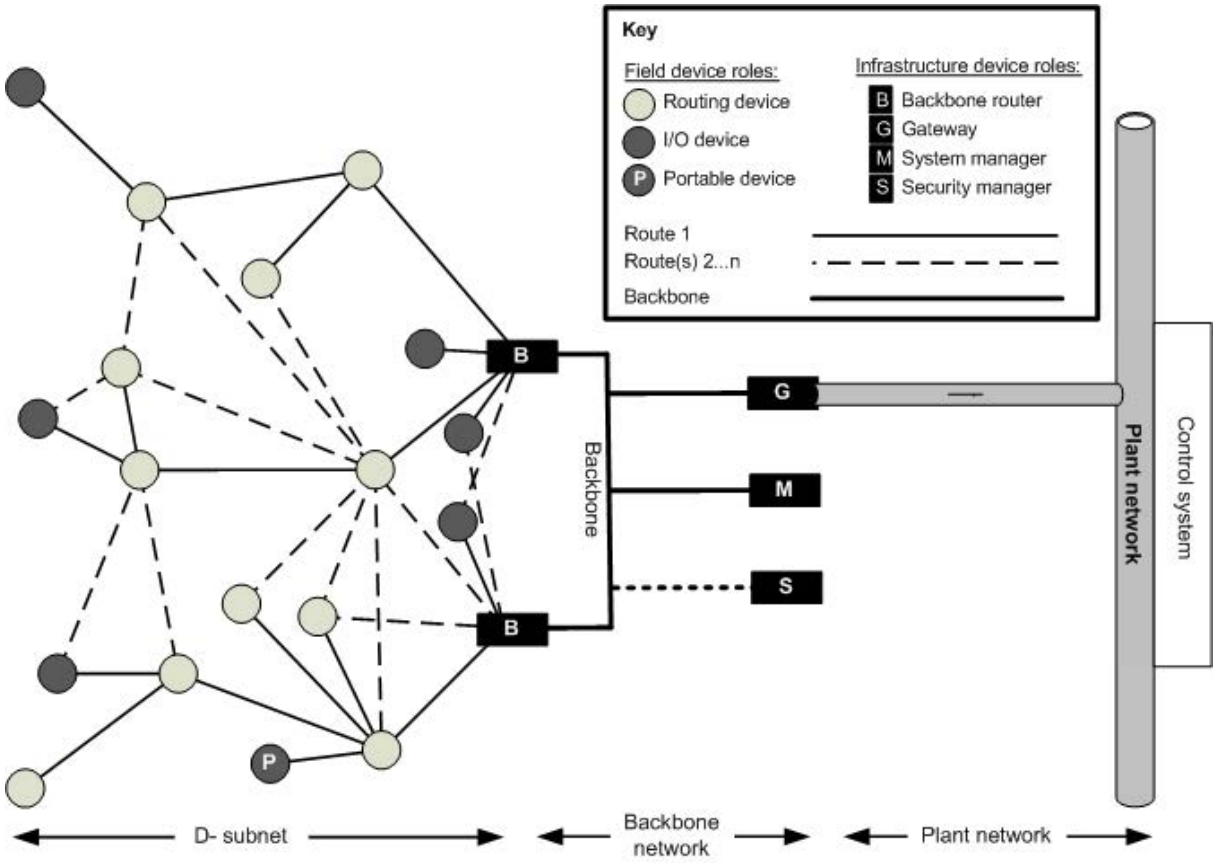


Figure 15 – Network with backbone – Device roles

The physically separated gateway, system manager, and security manager shown in Figure 15 can be implemented only in networks with a network backbone.

5.4 Protocol suite structure

The protocol layers for a device conforming to this standard are described in terms of the OSI Basic Reference Model, which is adapted as shown in Figure 16. All roles and device types compliant with this standard can be derived from this model by extension or restriction of common elements depicted in Figure 16.

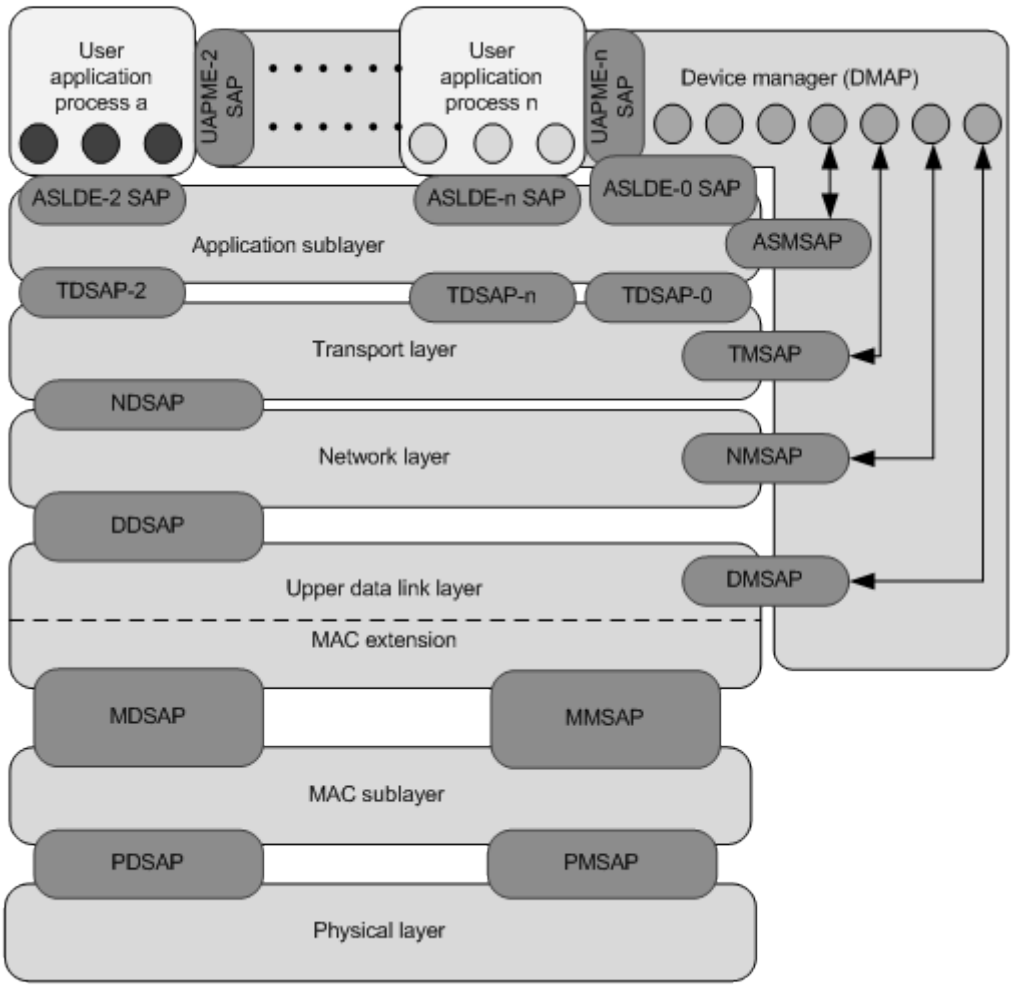


Figure 16 – Reference model used by this standard

As shown in Figure 16, each layer provides a service access point (SAP). The services of a layer are defined as the functions and capabilities of that layer that are exposed through the SAP to the surrounding layers. In general two types of SAPs are defined: data SAPs, which are used for operational data transfer, and management SAPs, which are used for layer management. The services provided by a layer are defined by the data flowing through the data SAPs and, in some cases, the states that a layer provides and the state transitions that are driven by the interaction across those SAP. The device manager is the entity within each device that performs the management function; in most cases it is accessed via a layer management SAP. The device manager has a dedicated path to several of the lower protocol layers within a device, to provide direct real-time control over the operation of those layers as well as direct access to diagnostics and status information.

In Figure 16 the device provides the functionality of each management SAP used by the DMAP for every protocol layer that is implemented.

Since compliance can be assessed only at external interfaces, including the content of data structures conveyed at those interfaces, all notional descriptions of how specific functionality could be implemented is necessarily only informative, not normative.

5.5 Data flow

5.5.1 General

The descriptions in 5.5 are intended to provide examples of how data may flow through the system. The set of examples is not intended to be exhaustive.

5.5.2 Native communications

A device communicates over the network using only ASL defined services as defined in Clause 12; the payloads are classified as either native or non-native. Native payloads are defined in Clause 12; non-native payloads are not defined within this standard.

5.5.3 Basic data flow

Figure 17 illustrates the steady-state data flow for a basic network compliant with this standard, such as the one shown in Figure 10.

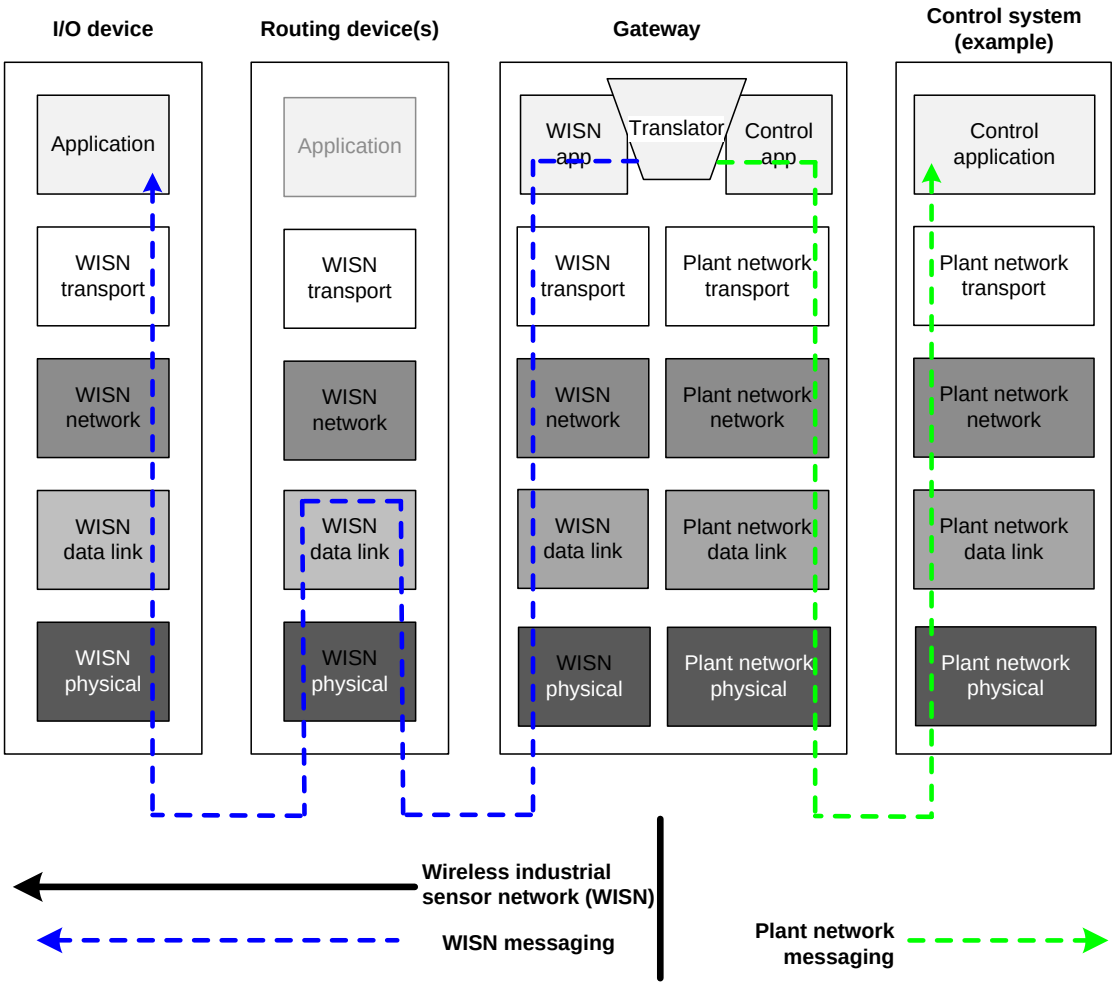


Figure 17 – Basic data flow

The I/O device is a sensor or actuator device within the D-subnet that contains physical, data-link, network, and transport layers as defined by this standard and runs an application that handles the sensor or actuator function.

The router routes messages on behalf of the I/O device. Routing within the D-subnet is performed entirely within the DL, and not within the NL (see Clause 9). In a real-world network, there will be one router for each additional hop between the device and the WISN-connected gateway or backbone router.

The gateway translates messages between the D-subnet and the plant network. The application running on the gateway consists of a component that communicates with the ALE of the I/O device, plus a component that communicates with an ALE within the control system, plus any components that facilitate translation between the two, such as a cache.

5.5.4 Data flow between I/O devices

Figure 18 illustrates the data flow for communication between I/O devices within a D-subnet. Routing within the D-subnet is performed entirely within the DL, and not within the NL.

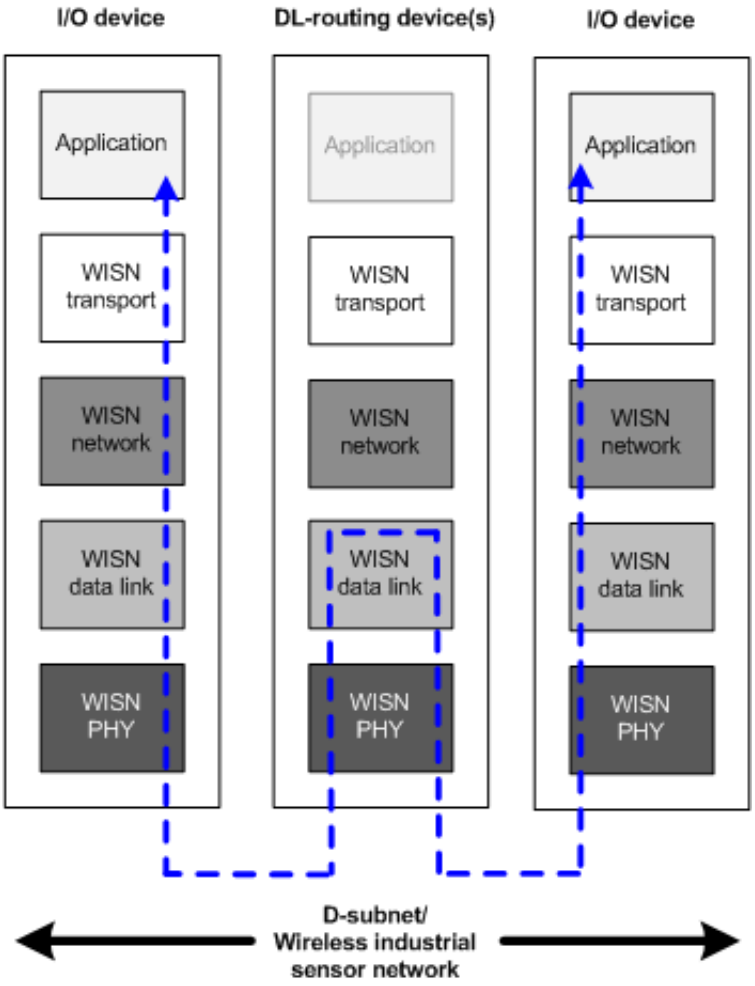


Figure 18 – Data flow between I/O devices

5.5.5 Data flow with legacy I/O device

Figure 19 illustrates a legacy I/O device that is integrated into a D-subnet via a legacy device adapter. An adapter is a specialization of a device that internally converts the protocol in the attached legacy device(s) to that of a network device compliant with this standard.

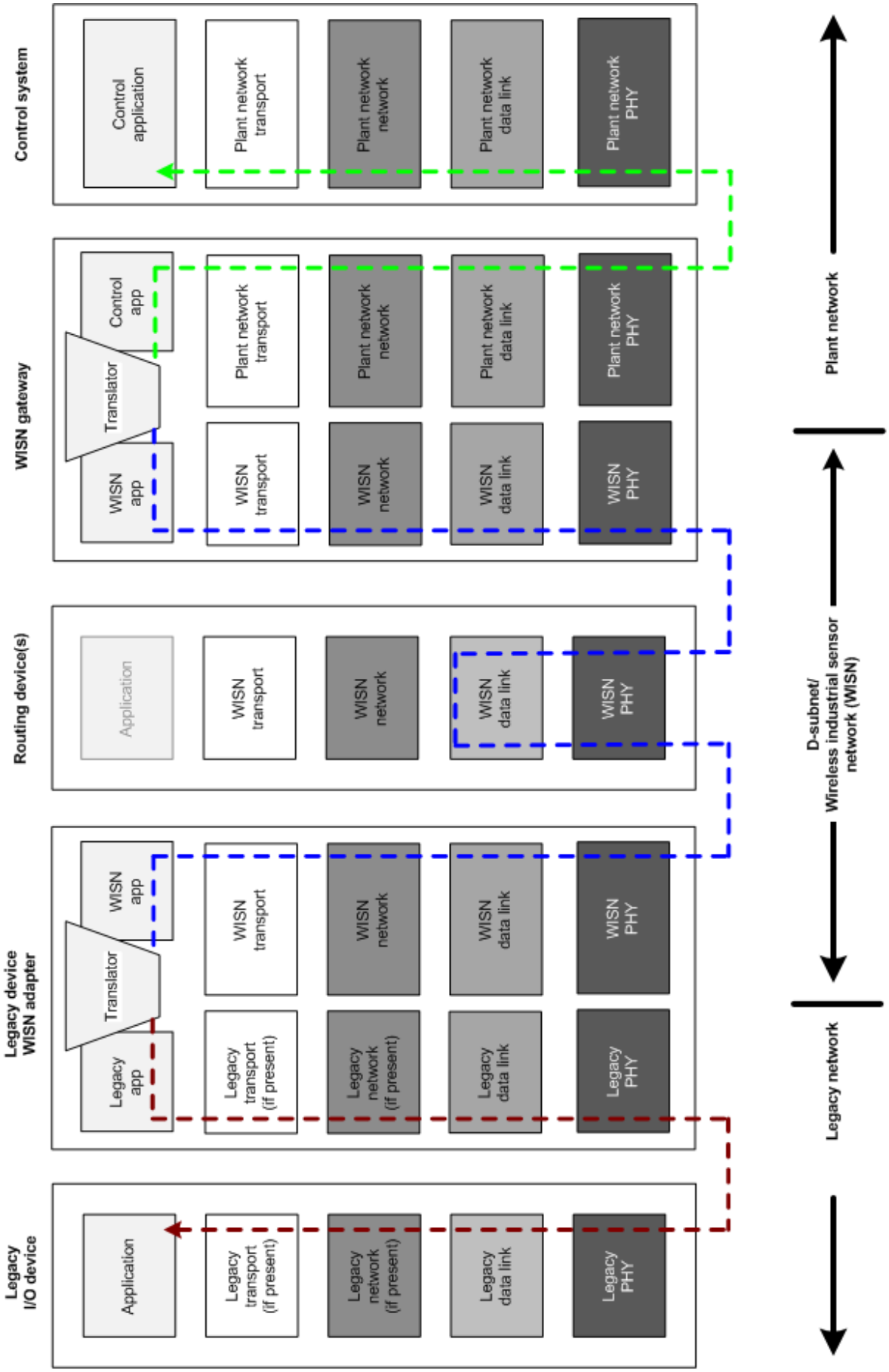


Figure 19 – Data flow with legacy I/O device

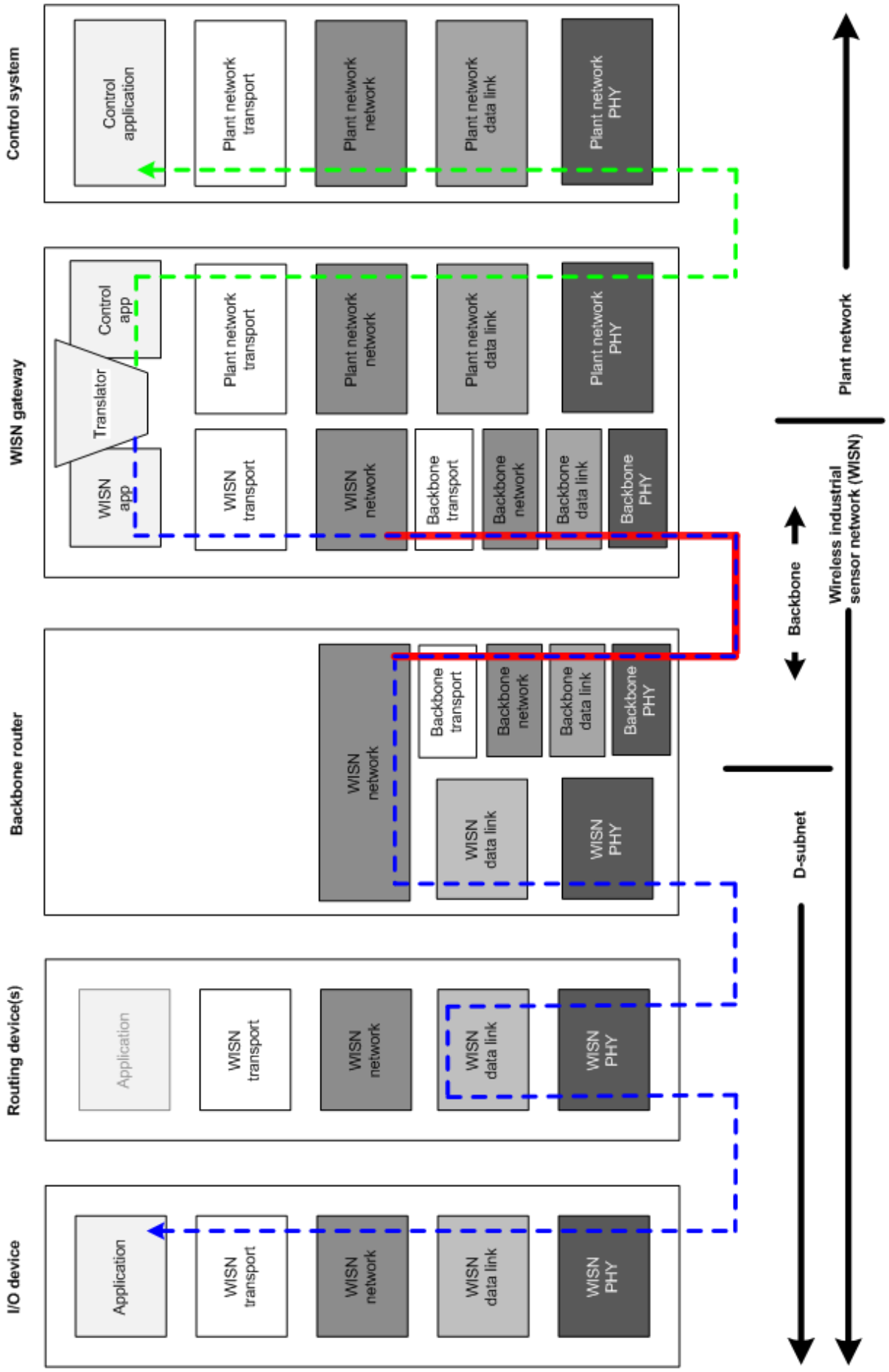


Figure 20 – Data flow with backbone-resident device

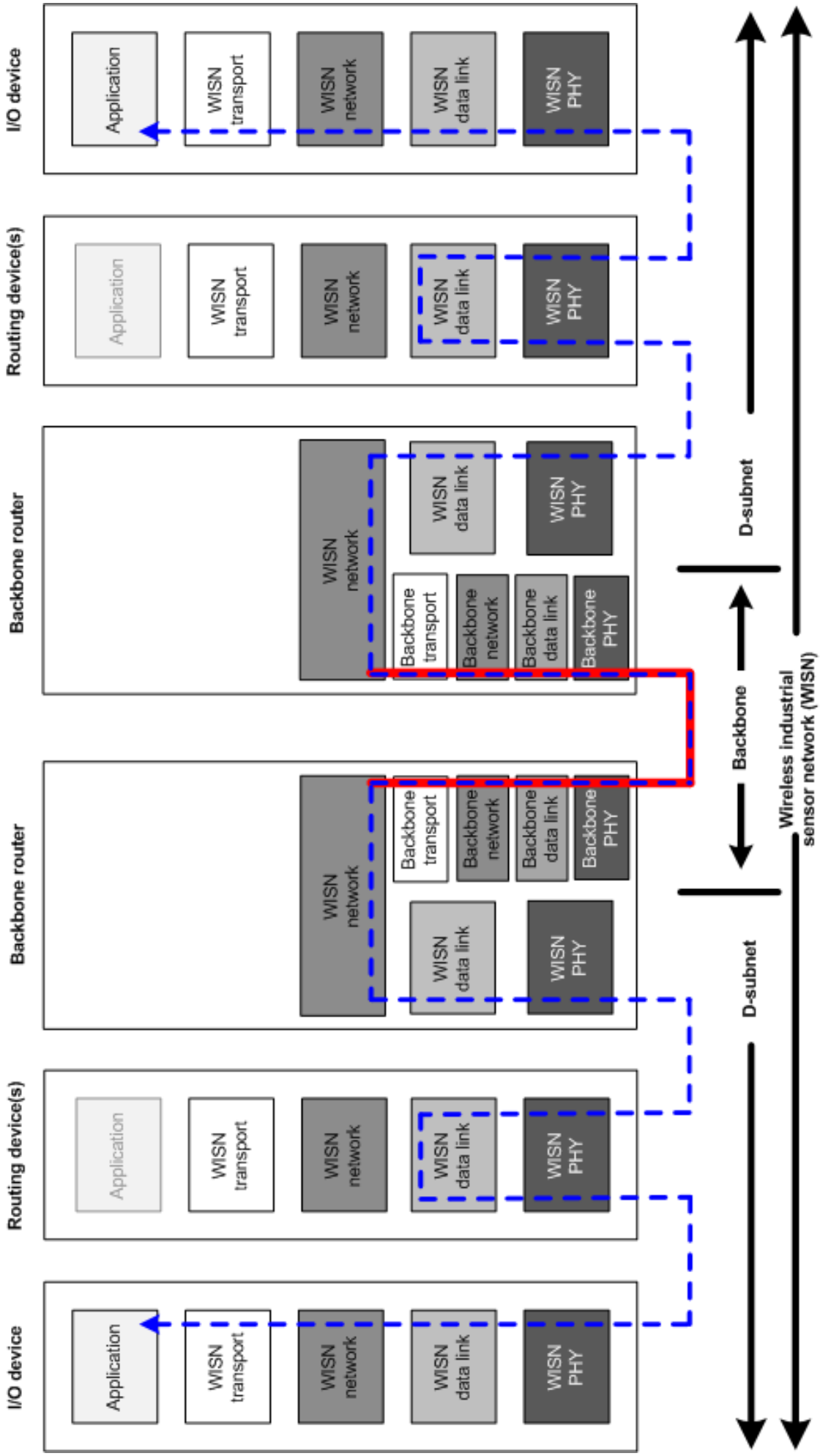


Figure 21 – Data flow between I/O devices via backbone subnet

5.5.6 Data flow with backbone

Figure 20 introduces a backbone router into the data flow.

The backbone router encapsulates NPDUs and relays them through backbone physical, data-link, network, and transport layers. The gateway uses the same backbone layers to recover the NPDUs. While it is not shown in Figure 20, the gateway may include both a PhLE and a DLE as defined by this standard, enabling the gateway to handle messages directly from a D-subnet, in addition to messages relayed through a backbone interface.

5.5.7 Data flow between I/O devices via backbone

Figure 21 illustrates how a backbone network handles standard-compliant message transfer when an I/O device communicates directly with another I/O device in a different D-subnet.

5.5.8 Data flow to a standard-aware control system or device

A standard-aware control system is a control system that understands messaging defined by this standard and does not need a gateway to perform protocol translation. Figure 22 illustrates data flow to a standard-aware control system.

NOTE Generic protocol translation is addressed in Annex O.

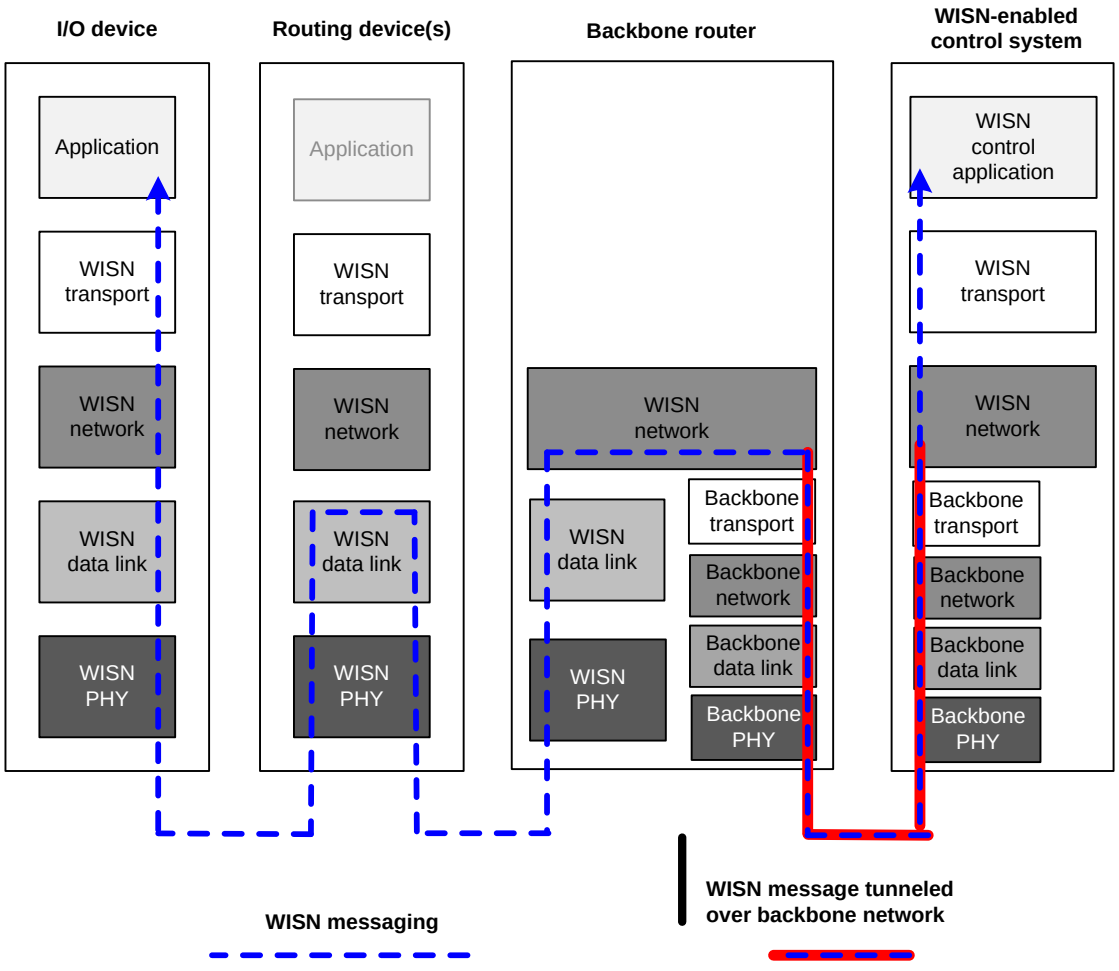


Figure 22 – Data flow to standard-aware control system

In general, for a device to be standard-aware, it needs only to support the application interface defined by this standard and to implement the application, transport, and

network layers defined by this standard. This makes it possible for two standard-aware devices to communicate via a plant network without using or requiring any sublayers.

5.6 Time reference

5.6.1 General

This standard's time is based on international atomic time (TAI) as the time reference; see 6.3.10. This standard's time is reported as elapsed seconds since the TAI instant of 00:00 on 1 January 1958 (i.e., 1958/01/01 00:00).

It is not possible or even desirable for every network to track an atomic clock precisely. Rather, every network shall have a sense of time that is

- monotonically increasing at a rate that closely matches real time;
- not to exceed an error of more than 1 s relative to the system time source; and
- delivered to various layers in field devices in consistent TAI units.

There are communication modes as defined in Clause 9 that require better than 1 s clock accuracy relative to the system time source.

For protocol operation, sequence of event reporting, and other purposes, time usually needs to be divided into increments of less than one second. For example, increments may be represented in multiple ways:

WISN clock ticks: Two octets to mark time in increments of 2^{-15} s (32 768 Hz, or ~30,52 μs per tick).

Microsecond precision: Three octets to mark time in increments of 2^{-20} s (~0,95 μs per increment).

Nanosecond precision: Four octets to mark time in increments of 2^{-30} s (~0,93 ns per increment).

Devices needing to convert TAI time to hh:mm:ss format, such as on a user display, may account for a coordinated universal time (UTC) accumulated-leap-second adjustment. This adjustment is available to field devices from the system manager. If the UTC adjustment is used by a field device, it should refresh the adjustment at the start of each month.

NOTE A list of such UTC adjustments is maintained at ftp://maia.usno.navy.mil/ser7/tai-utc.dat .

Simultaneous UTC update requests by many devices may cause a storm of activity in the DL. This should be considered in the DMAP design; its avoidance is not covered by the current DL specification.

All devices in a network share the TAI time reference with varying degrees of accuracy. Each device within a network shall maintain time accurately to within 1 s.

The system manager directs devices on the system to a device implementing the role of system time source. In most cases, this device will also be filling the system manager role. However, the time-source responsibility can be redirected to any device with a more capable source of time.

The gateway shall be responsible for converting between nominal network TAI time and an external non-TAI time reference if one is being used.

For more information on the requirements for the time source, see 6.3.10.

5.6.2 Time synchronization

To propagate host time, a gateway may periodically synchronize the time sense in an attached D-subnet to an external time source by requesting time changes via DLMOs.

The WISN provides time synchronization for applications so that, at the device level, they may use time to coordinate activities or to time-stamp data, an activity that could improve energy use and enhance reliability. System time shall be available from at least one device (a system time source) on each WISN. See Clause 9.

5.7 Firmware upgrades

The overall system, and each device on the WISN, shall provide the capability of upgrading device firmware that implements this standard via the wireless network (see 6.3.6). The system shall support a common mechanism, such as a time-based trigger, to inform all devices to switch concurrently to the new firmware; this mechanism may be used to minimize the number of devices that are left stranded with an incompatible network protocol suite. The security mechanisms built into this standard are used during a firmware upgrade.

Each version of the protocol shall support previous versions to the extent necessary to support upgrading firmware via the wireless network.

5.8 Wireless backbones and other infrastructures

Devices compliant with this standard are managed devices. All devices compliant with this standard shall implement the device management interfaces at each layer, but they may implement only the functionality of their required functional layers.

The system supports both wired and wireless backbone networks through the use of backbone routers. The operation of backbone networks is not addressed by this standard.

More information on backbone networks and their implied characteristics can be found in Annex E.

6 System management role

6.1 General

6.1.1 Overview

The system management role supports network management of the network as a whole, as well as device management of the devices operating within the network. Network management includes management of the various communications resources across the network and across all protocol layers of the architecture. Device management supports localized management of the communications resources, and potentially other resources, of a device.

The management functions described by this standard support

- joining the network and leaving the network;
- reporting of faults that occur within the network;
- communication configuration;
- configuration of clock distribution and the setting of system time;
- device monitoring;
- performance monitoring and optimization; and
- security configuration and monitoring.

6.1.2 Components and architecture

The primary components of the management service include a device management application process (DMAP) that resides on every device compliant with this standard, as well as a system management application process (SMAP) that shall reside on a device that implements the system manager role. Roles are described in 5.2.6.5. The DMAP is a special type of user application process (UAP) that is dedicated to managing the device and its communications services, described in 12.4.3 and 12.4.4. The DMAP and the system manager shall be capable of communicating with each other over the network using the standard-defined application sublayer services, and shall together provide a means to access management information remotely and to manage the system and its devices. System management is accomplished via inter-device messaging, while device management is accomplished by local intra-device communications.

The management architecture of this standard is shown in Figure 23.

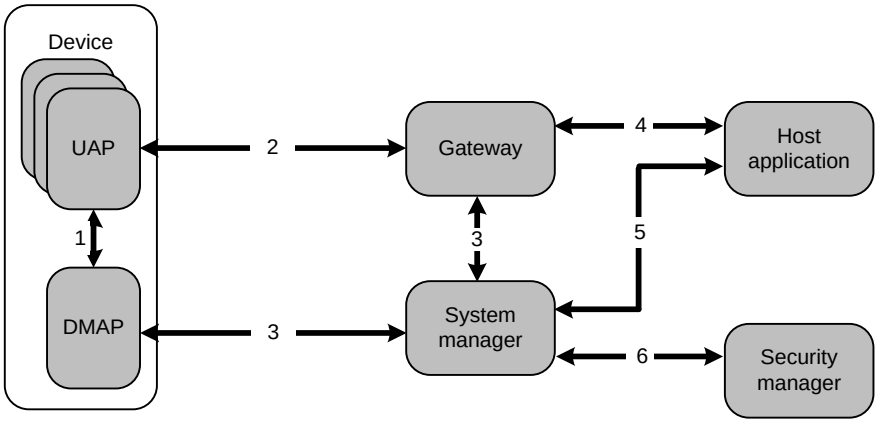


Figure 23 – Management architecture

Devices compliant with this standard shall be managed through two distinct classes of application processes, UAPs and the DMAP. The UAPs are configured and monitored by host applications, such as automated management systems, or by host proxy applications in the gateways. The DMAP in the device shall be managed by the system manager.

Figure 23 shows the management model relationships of this standard. For the paths illustrated in Figure 23, this standard provides a normative description of the communication protocols for paths 2 and 3. Communication protocols for paths 1, 4, 5 and 6 are informative examples of implementations in this standard.

This standard defines the system management communication protocols that shall be used to control and monitor the DMAPs in the network and the relevant communication paths. In this case these communications travel on path 3 in Figure 23.

The system manager includes communication paths outside of the standard-compliant network that allow other devices to interact with it. In Figure 23, path 5 shows a connection between the system manager and the host application. This path enables the host application to retrieve network status and request network services. The system manager also communicates with the security manager over path 6 to configure security in the network and to report status.

The user applications on devices compliant with this standard communicate with gateways and host applications using the standard protocols shown over path 2 in Figure 23. This is described in Clause 12 and Annex U.

The plant-based host application communicates with the gateway using plant protocols that travel over path 4. The system manager does not communicate directly with the UAPs. There is an intra-device communication path that enables the DMAP and UAP processes to interact via the intra-device communication path 1 in Figure 23 between the system manager and a gateway. This is performed over a virtual interface 1 in Figure 23, UAPME-SAP, using the UAP management object (UAPMO) which is described in 12.15.2.2.

6.1.3 Management functions

Every network that is compliant with this standard shall include at least one system management role and one security management role. These roles shall be accessible to all standard compliant devices on this network.

System management is a specialized role that governs the network, the operation of devices on the network, and network communications. The functions defined within this role are performed by the system manager, providing policy-based control of the runtime communication configuration. The system manager monitors and reports on communication configuration, performance, fault conditions, and operational status. This is described in 6.3.7. The system manager also provides time-related services. Some system management functions may be completely automated, while others may be human-assisted.

The system manager supports configuration of the standard-compliant network, including attributes of the protocol suite from DL to AL for system management applications. It manages the establishment, modification and termination of contracts that are used by devices compliant with this standard to communicate with each other. The functions of the system manager do not include the control, configuration, and monitoring of the UAPs on the device. These management functions are controlled by host applications on plant networks or in handheld maintenance tools.

Security management of the system is a specialized function that is realized in one entity and that works in conjunction with the system management function to enable secure system operation. This function is performed by the security manager. Some system security management functions may be completely automated, while others may be human-assisted.

Every device compliant with this standard shall contain a DMAP. The DMAP includes a local device security management function. The DMAP cooperates with the system manager and the security manager to enable the usage of system resources by the device and the secure management of the resources of a device. For example, the DMAP may ask to join the network, ask for communication bandwidth, request a communication configuration, and report its health. The system manager and the security manager authorize the device to join the network, allocate communication bandwidth, configure the device, and collect health reports. These health reports are stored in the system manager and are used to make communication configuration decisions.

In order to compartmentalize security functions, the management architecture defined by this standard supports separable system management and security management functions at both the system and device levels. Thus, the security manager is logically separable from the system manager. More details about the security manager are provided in Clause 7.

NOTE The system management and security management functions often are included within a single physical entity.

6.2 DMAP

6.2.1 General

The DMAP is a special type of application process dedicated to managing the standard-compliant device and its communications services. A DMAP resides on every device compliant with this standard.

6.2.2 Architecture of device management

As shown in Figure 16, the protocol suite structure of a device includes the networking protocol layers and the UAPs.

The DMAP is shown in relation to the other protocol suite components on the right side of the protocol suite structure, including arrows depicting access to the management SAPs for several of the protocol layers. The components within the DMAP are modeled as objects, known as management objects, which have features that are accessible over the network. The DMAP, like all application processes, is able to use the application sublayer to communicate. The DMAP shall use the application-sublayer SAP ASLDE-0 SAP for normal data communications. This application sublayer SAP shall correspond to TL SAP TDSAP-0 which shall correspond to port number 0xF0B0. The application sublayer provides communication services to enable the objects within the DMAP to interact with the system manager over the network. These communication services are described in 12.17.

6.2.3 Definition of management objects

The objects defined in the DMAP follow the specification that is used to define UAP objects. The templates for defining object types, object attributes, object methods and object alerts are specified in Annex I.

The management objects are extensible by device manufacturers and network protocol suite/device developers. This is described in 12.5. Attribute and method identification space is set aside for manufacturer-defined device-specific objects. The system manager shall not be required to implement support for proprietary extensions for that device to interoperate and perform its primary function.

6.2.4 Management objects in DMAP

The DMAP shall contain a number of management objects that support device management operations. These objects shall collectively perform two types of device management functions. First, these objects shall manage the device locally by manipulating attributes and invoking methods on layer management SAPs. Second, the management objects shall be accessible remotely using the ASL services such that a system manager may manipulate attributes and invoke methods on the device management objects or capture alerts from the objects. These objects are conceptual in that there are no object-oriented implementation requirements in the device, except that the externally visible behavior in terms of over-the-air ASL messaging shall be consistent with the model of object communications having the specified attributes, methods, and alerts.

As shown in Figure 24, the DMAP shall include a set of layer management objects, a device management object, a device security management object, an alert reporting management object, an upload/download object, and other management objects.

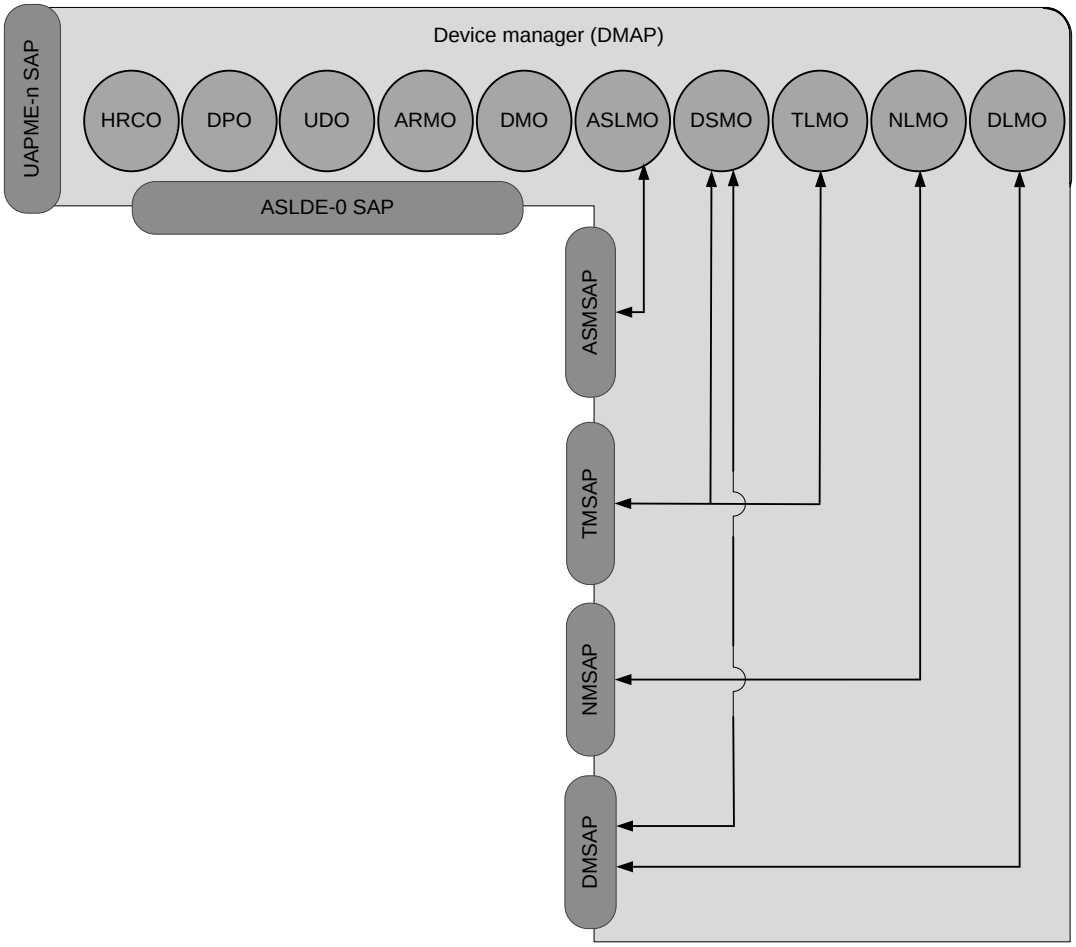


Figure 24 – DMAP

The standard management objects defined in this standard are given in Table 1.

Table 1 – Standard management object types in DMAP

Standard object type name	Standard object type identifier	Standard object identifier	Object description
Device management object (DMO)	127	1	This object facilitates the management of the general device-wide functions of the device; see 6.2.7.1
Alert reporting management object (ARMO)	126	2	This object facilitates the management of the alert reporting functions of the device; see 6.2.7.2
Device security management object (DSMO)	125	3	This object facilitates the management of the security functions of the device; see 6.2.7.5
DL management object (DLMO)	124	4	This object facilitates the management of a device DLE; see 6.2.8.2.2
NL management object (NLMO)	123	5	This object facilitates the management of a device NLE; see 6.2.8.2.2.5
TL management object (TLMO)	122	6	This object facilitates the management of a device TLE; see 6.2.8.2.2.6
Application sublayer management object (ASLMO)	121	7	This object facilitates the management of the device ALE; see 6.2.8.2.2.8

Standard object type name	Standard object type identifier	Standard object identifier	Object description
Upload/download object (UDO)	3	8	This object facilitates the management of the upload/download functions of the device; see 6.2.7.3
Device provisioning object (DPO)	120	9	This object facilitates the provisioning of the device before it joins a D-subnet; see 6.2.7.6
Health reports concentrator object (HRCO)	128	10	This object facilitates the periodic publication of device health reports to the system manager; see 6.2.7.7
Reserved for future editions of this standard	119..114	—	—

6.2.5 Communications services provided to device management objects

The application level services provided to the DMAP objects are the same as those provided by the application sublayer to UAP objects. These services include client/server (C/S), publish/subscribe (publish/subscribe), source/sink (source/sink), and alert reporting (AR). Details of these services, provided by the application sublayer, are given in 12.17.

As shown in Figure 25, TDSAP-0, which corresponds to port number 0xF0B0, shall be used for accessing the management objects in the DMAP, which in turn access the layer management attributes through the layer management SAP.

Access to the device management objects is protected by the TL security mechanisms described in 11.3.

Access to the DMAP objects is restricted to the SMAP with the following exceptions:

- The ARMO can also be accessed by alert masters that receive alerts originating in the device. See 6.2.7.2.3.
- A joining device is allowed to access the device management object methods used during the subnet joining process. See 6.3.9.2.2.

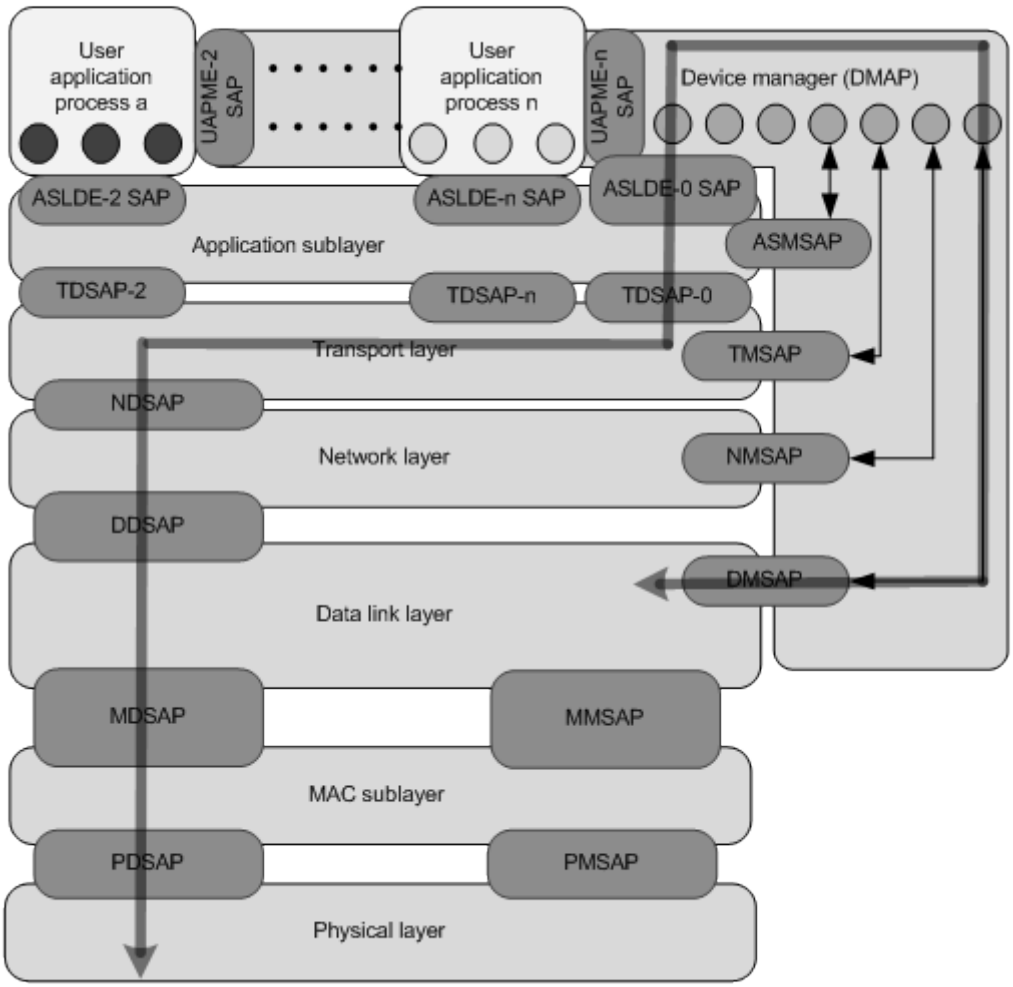


Figure 25 – Example of management SAP flow through standard protocol suite

Client/server interactions (including reading and writing of attributes, executing of methods, joining, and requesting and providing contracts) are the primary tools used for system management. In addition, the DMAP may use the alert reporting services of the ASL to report to the system manager when certain management-related conditions are detected. Designers of management objects may use alerts at various priority levels to help accomplish system and device management functions.

6.2.6 Attributes of management objects

6.2.6.1 General

The layer management SAPs shown in Figure 25 provide access to the management information in the various layers of the protocol suite.

This information is represented by attributes defined in the management objects of the DMAP, which can be monitored and operated on by the system manager. Details of the management objects are given in 6.2.3. The attributes in the layer management objects are used to configure the protocol layers and to monitor their status. The template for describing the attributes in all management objects is provided in Annex I, for use in proprietary extensions and future editions of this standard.

Attributes shall have a data type that is either a standard-defined scalar type or a standard-defined data structure. More details about attributes are given in 12.6.2.

A structured attribute is a special type of attribute that has a data type consisting of an array of standard-defined data structures. The array model is used to permit object access through indexing, where the index is the key attribute for access to the object.

Management information that needs to be visualized as a collection of one or more tables is modeled as structured attributes defined in the management objects.

Attributes defined in management objects can be accessed using the standard ASL-provided read or write services. Such operations enable configuration of each layer and monitoring of its status. They can be used to retrieve, set/modify, and reset the values of attributes. Operations on attributes are described in Annex J.

6.2.6.2 Structured attribute index field

Since structured attributes are described as arrays of data structures, one or more index fields for such arrays need to be indicated in the definition of each such structured attribute. This is done by including an * (asterisk) after the element name(s) in the table describing the data structure. The template for defining a data structure is given in Annex I.

6.2.6.3 Metadata of structured attribute

Structured attributes represent information tables. To provide external access to the number of objects in, and the capacity of, any such table, additional meta-attributes that contain such information are defined for management objects. Such attributes represent the metadata of the corresponding structured attributes.

The standard data type for a metadata attribute is given in Table 2.

Table 2 – Metadata_attribute data structure

Standard data type name: Metadata_attribute		
Standard data type code: 406		
Element name	**Element identifier**	**Element type**
Count (number of indexed rows currently in the attribute)	1	Type: Unsigned16 Classification: Static Accessibility: Read only
Capacity (number of rows that the attribute can hold)	2	Type: Unsigned16 Classification: Static Accessibility: Read only

6.2.7 Definitions of management objects in DMAP

6.2.7.1 Device management object

As shown in Figure 24, the DMAP includes a set of management objects. The device management object (DMO) in the DMAP shall provide access to attributes having device-wide scope. Attributes of the DMO shall include the primary EUI64Address of the DLE, a vendor ID, a serial number, an identification of the current revision of the communications software, and the device's power source class. More details about DMO are provided in 6.2.8.1.

6.2.7.2 Alert reporting management object

6.2.7.2.1 General

The alert reporting management object (ARMO) is used to manage all the alert reports of the device. **Alert** is the term used to describe the action of reporting an event condition or an alarm condition. **Event** is the term for a transient (i.e., stateless) condition, used to report when something happened. **Alarm** is the term used for a condition that maintains the state until the condition clears, which is reported on change of state. Alerts, including events and alarms, are envisioned to be of high utility for managing a network compliant with this standard.

There shall be at most one ARMO per device. Both alarms and events shall be reported through the ARMO. When an alert is triggered, it indicates a significant situation that needs to be reported. The ARMO shall encapsulate the report, handle timeouts and retries, and throttle alert reporting from the device.

The ARMO functions as an alert proxy for the objects present in the device. All alerts generated by any object present in a device shall be sent only by the ARMO which is a management object that is part of the DMAP. The Alert data APDU shall indicate in its APDU header that the originator of the communication is the ARMO object and the DMAP of the device reporting the alert. The object and UAP that originated the actual alert APDU shall be identified in the content of the Alert report rather than in the APDU headers.

Each alert shall be acknowledged by the device receiving the alert. Each alert acknowledgment shall be addressed to the ARMO of the device that originated the alert. Alerts are reported promptly and time-stamped accurately using queued alert reporting. Queued alert reporting involves the alert detecting device reporting the condition using a source/sink communication flow and receiving an ACK/NAK DPDU in return.

NOTE The intent of specifying the ARMO in this standard is to separate alert detection from the management of reporting the alert condition. Unlike some wire-oriented legacy protocols, this standard consolidates alerts locally in order to minimize externalized messaging and energy consumption.

The alert model used in this standard is described in 12.8.

The interfaces between the ARMO and all other objects, both in UAPs and in the DMAP, are device internal and are not specified in this standard.

6.2.7.2.2 Alert types

Alert classes, alert directions, and alert priorities are defined in 12.11. The alert category indicates whether the alert is a device diagnostic alert, a communication diagnostic alert, a security alert, or a process alert. The alert type provides additional information regarding the alert, specific to the alert category and specific to the application object generating the alert.

Table 3 provides the alert types for the alert categories of the communication diagnostic alert category. Table 4 provides the alert types for the security alert category. Table 5 provides the alert types for the device diagnostic alert category. Table 6 provides the alert types for the process alert category.

Table 3 – Alert types for communication diagnostic category

Alert type	Alert category: Communication diagnostic					
	ARMO	ASLMO	DLMO	NLMO	TLMO	DMO
0	Alarm_Recovery_Start; see Table 8	Malformed APDUCommunicationAlert; see 12.19.5	DL_Connectivity; see 9.6.1	NL Dropped PDU; see 10.4.3	IllegalUseOfPort; see 11.6.2.5.4	Device_Power_Status_Check; see 6.2.8.1.2
1	Alarm_Recovery_End; see Table 8	—	Neighbor Discovery; see 9.6.2	—	TPDUonUnregisteredPort; see 11.6.2.5.4	Device_Restart; see 6.2.8.1.2
2	—	—	—	—	TPDUoutOdSecurityPolicies; see 11.6.2.5.4	—

Table 4 – Alert types for security alert category

Alert type	Alert category: Security		
	ARMO	DSMO	DPO
0	Alarm_Recovery_Start; see Table 8	Security_MPDU_Fail_Rate_Exceeded; see 7.11.4	Not_On_Whitelist_Alert; see Table 374
1	Alarm_Recovery_End; see Table 8	Security_TPDU_Fail_Rate_Exceeded; see 7.11.4	Inadequate_Join_Capability_Alert; see Table 374
2	—	Security_Key_Update_Fail_Rate_Exceeded; see 7.11.4	—

Table 5 – Alert types for device diagnostic alert category

Alert type	Alert category: Device diagnostic
	ARMO
0	Alarm_Recovery_Start; see Table 8
1	Alarm_Recovery_End; see Table 8

Table 6 – Alert types for process alert category

Alert type	Alert category: Process				
	ARMO	AI	AO	BI	BO
0	Alarm_Recovery_Start; see Table 8	See 12.19.7	See 12.19.7	See 12.19.7	See 12.19.7
1	Alarm_Recovery_End; see Table 8	—	—	—	—

6.2.7.2.3 Alert master

Alerts shall be sent to alert-receiving objects. Alert-receiving objects are defined in 12.15.2.3. Each alert category may have a different alert-receiving object residing in a different device. Devices that receive these alerts are known as alert masters.

DMAP access is often restricted to the SMAP present in the system manager. In an exception to this general principle, alert masters are allowed to access the ARMO object present in the DMAP. DMAP access by alert masters shall be limited to the ARMO, unless the alert master

uses the DMAP-SMAP session established when the device joined the network. The alert masters to which the device is configured to send alerts are listed in Table 7.

6.2.7.2.4 Alert queue

Alerts belonging to each category are assumed to be placed into an internal queue provided per-category in the device. Both types of alerts, events (stateless) and alarms (stateful) will be placed in the same queue, filtered by category. The queue is necessary to provide a guaranteed delivery of alerts to the alert master. Every alert to be reported to the alert master is placed into this reporting queue.

The size of the queue should be big enough to accommodate all events as well as all possible alarm conditions simultaneously in order to support alarm recovery without losing any alarms.

Although placed in the same queue, events and alarms will be prioritized differently. The device shall report an event with higher priority before an event with lower priority. For alarms, the queue is emptied sequentially; the oldest alarm is reported first. When the queue is full and a new alarm is submitted, the oldest alarm is dropped from the queue regardless of its reporting state.

6.2.7.2.5 Alert state models

The state tables and transitions for alarms and events are given in 12.9 and 12.10.

6.2.7.2.6 Alarm recovery

It is often useful to be able to recover all alarms currently active within a device. The need for alarm recovery arises whenever a connection to the device is lost for a period of time or whenever an alert master commands an alarm recovery.

Alarm recovery consists of the following set of activities:

- The alert master commands an alarm recovery by using the Alarm_Recovery method of the ARMO. This method is described in Table 9.
- The ARMO sends a recovery start alert to the alert master, which indicates that the ARMO has received a command to recover alarms and those active alarms will follow.

NOTE The process for re-sending these active alarms within a device is not specified.

- The ARMO sends a recovery end alert to the alert master.

The ARMO is responsible for generating alarm recovery start and end alerts and for coordinating the alarm recovery process with the application objects residing within the device.

6.2.7.2.7 Alert reporting management object attributes, alerts and methods

The attributes of the ARMO are defined in Table 7.

Table 7 – ARMO attributes *(1 of 3)*

Standard object type name: Alert reporting management object (ARMO)				
Standard object type identifier: 126				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Alert_Master_Device_ Diagnostics	1	Alert master for alerts that belong to the device diagnostics category	Type: Alert communication endpoint Classification: Static Accessibility: Read/write Valid range: See 12.16.3.5	Typically set to a gateway for the device's information, but can be changed to any other standard-compliant device with a valid IPv6Address [a]
Confirmation_Timeout_ Device_Diagnostics	2	Timeout waiting for acknowledgment of a device diagnostic alarm that was sent to the alert master	Type: Integer16 Classification: Static Accessibility: Read/write Default value: 10	Timeout independent of proximity to alert master. A value of $N > 0$ specifies a duration of N s, while $N < 0$ specifies a duration of -1/N s. $N = 0$ is not permitted [b]
Alerts_Disable_Device_ Diagnostics	3	Command to disable/enable all device diagnostics alerts	Type: Boolean8 Classification: Static Accessibility: Read/write Default value: FALSE	FALSE = enable, TRUE = disable
Alert_Master_Comm_ Diagnostics	4	Alert master for alerts that belong to the communication diagnostics category	Type: Alert communication endpoint Classification: Static Accessibility: Read/write Valid range: See 12.16.3.5	Typically set to be the system manager for the device, but can be changed to any other standard-compliant device with a valid IPv6Address; the device shall set this to be its system manager after it joins the network; see 6.3.7.2
Confirmation_Timeout_ Comm_Diagnostics	5	Timeout waiting for acknowledgment of a communication diagnostic alarm that was sent to the alert master	Same as attribute 2	Same as attribute 2
Alerts_Disable_Comm_ Diagnostics	6	Command to disable/enable all communication diagnostic alerts	Type: Boolean8 Classification: Static Accessibility: Read/write Default value: FALSE	FALSE = enable, TRUE = disable

Table 7 *(2 of 3)*

Standard object type name: Alert reporting management object (ARMO)				
Standard object type identifier: 126				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Alert_Master_Security	7	Alert master for alerts that belong to the security category	Type: Alert communication endpoint Classification: Static Accessibility: Read/write Valid range: See 12.16.3.5	Typically set to be the system/security manager for the device, but can be changed to any other standard-compliant device with a valid IPv6Address. The device shall set this to be its security manager after it joins the network; see 6.3.7.2
Confirmation_Timeout_Security	8	Timeout waiting for acknowledgment of a security alarm that was sent to the alert master	Same as attribute 2	Same as attribute 2
Alerts_Disable_Security	9	Command to disable / enable all security alerts	Type: Boolean8 Classification: Static Accessibility: Read/write Default value: FALSE	FALSE = enable, TRUE = disable
Alert_Master_Process	10	Alert master for alerts that belong to the process category	Type: Alert communication endpoint Classification: Static Accessibility: Read/write Valid range: See 12.16.3.5	Typically set to a gateway for the device's information, but can be changed to any other standard-compliant device with a valid IPv6Address [a]
Confirmation_Timeout_Process	11	Timeout waiting for acknowledgment of a process alarm that was sent to the alert master	Same as attribute 2	Same as attribute 2
Alerts_Disable_Process	12	Command to disable / enable all process alerts	Type: Boolean8 Classification: Static Accessibility: Read/write Default value: FALSE	FALSE = enable, TRUE = disable

Table 7 *(3 of 3)*

Standard object type name: Alert reporting management object (ARMO)				
Standard object type identifier: 126				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Comm_Diagnostics_Alarm_ Recovery_AlertDescriptor	13	Used to change the priority of alarm recovery start and end events (described in Table 8) that belong to the communication diagnostics category; these events can also be turned on or turned off	Type: Alert report descriptor Classification: Static Accessibility: Read/write Default value: [FALSE, 3] Valid range: See 12.16.3.7	—
Security_Alarm_Recovery_ AlertDescriptor	14	Used to change the priority of alarm recovery start and end events (described in Table 8) that belong to the security category; these events can also be turned on or turned off	Type: Alert report descriptor Classification: Static Accessibility: Read/write Default value: [FALSE, 3] Valid range: See 12.16.3.7	—
Device_Diagnostics_Alarm_ Recovery_AlertDescriptor	15	Used to change the priority of alarm recovery start and end events (described in Table 8) that belong to the device diagnostics category; these events can also be turned on or turned off	Type: Alert report descriptor Classification: Static Accessibility: Read/write Default value: [FALSE, 3] Valid range: See 12.16.3.7	—
Process_Alarm_Recovery_ AlertDescriptor	16	Used to change the priority of alarm recovery start and end events (described in Table 8) that belong to the process category; these events can also be turned on or turned off	Type: Alert report descriptor Classification: Static Accessibility: Read/write Default value: [FALSE, 3] Valid range: See 12.16.3.7	—
Reserved for future editions of this standard	17..63	—	—	—

[a] This information is expected to be configured by the host application after the device joins the network.

[b] All alarms require acknowledgement.

The alerts of the ARMO are defined in Table 8.

Table 8 – ARMO alerts

Standard object type name(s): Alert reporting management object (ARMO)				
Standard object type identifier: 126				
Description of the alert: Alarm recovery begin and end events for alarms that belong to all categories				
Alert class (Enumerated: alarm or event)	**Alert category (Enumerated: device diagnostic, comm. diagnostic, security, or process)**	**Alert type (Enumerated: based on alert category)**	**Alert priority (Enumerated: urgent, high, med, low, journal)**	**Description of value included with alert**
0 = Event	1 = Comm. diagnostics	0 = Alarm_Recovery_Start	3 = Low	Generated by the ARMO for the comm. diagnostics alert master indicating that the alarm recovery command has been received; all outstanding communication diagnostics alarms are reported after this event is raised
0 = Event	1 = Comm. diagnostics	1 = Alarm_Recovery_End	3 = Low	Generated by the ARMO for the comm. diagnostics alert master indicating that the alarm recovery process has ended
0 = Event	2 = Security	0 = Alarm_Recovery_Start	3 = Low	Generated by the ARMO for the security alert master indicating that the alarm recovery command has been received; all outstanding security alarms are reported after this event is raised
0 = Event	2 = Security	1 = Alarm_Recovery_End	3 = Low	Generated by the ARMO for the security alert master indicating that the alarm recovery process has ended
0 = Event	0 = Device diagnostics	0 = Alarm_Recovery_Start	3 = Low	Generated by the ARMO for the device diagnostics alert master indicating that the alarm recovery command has been received; all outstanding device diagnostics alarms are reported after this event is raised
0 = Event	0 = Device diagnostics	1 = Alarm_Recovery_End	3 = Low	Generated by the ARMO for the device diagnostics alert master indicating that the alarm recovery process has ended
0 = Event	3 = Process	0 = Alarm_Recovery_Start	3 = Low	Generated by the ARMO for the process alert master indicating that the alarm recovery command has been received; all outstanding process alarms are reported after this event is raised
0 = Event	3 = Process	1 = Alarm_Recovery_End	3 = Low	Generated by the ARMO for the process alert master indicating that the alarm recovery process has ended

The method of the ARMO used to recover alarms of the different categories shall be as defined in Table 9.

Table 9 – Alarm_Recovery method

Standard object type name(s): Alert reporting management object (ARMO)				
Standard object type identifier: 126				
Method name	**Method ID**	**Method description**		
Alarm_Recovery	1	Method to recover alarms that belong to the category mentioned in the input argument		
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Alert_Category	Data type: Unsigned8	Named values: 0: device diagnostics 1: comm. diagnostics 2: security 3: process
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	—	—	—	—

6.2.7.3 Upload/download object

The attributes, methods and state machines of the UDO in the DMAP shall be as per the definition given in 12.15.2.4. The object identifier of the UDO in the DMAP shall be 8.

An upload/download object (UDO) is used for uploading or downloading large blocks of information to/from a device. The UDO may be used to support downloading a new version of communications firmware or data. The UDO maintains revision control information. The UDO is described in 12.15.2.4.

The firmware upgrade process used by the system manager for over-the-air firmware upgrades is described in 6.3.6. The methods and attributes of the upload/download object in the DMAP of the device can be used for sending firmware updates to the device.

The firmware upgrade process may include a cut-over mechanism that specifies a cut-over time (after the update is delivered), at which point devices begin using the new firmware. The CutoverTime attribute in the UDO shall be used to indicate this cut-over time. The cut-over time uses the shared sense of time configured by the system manager.

Support is provided for vendor-specific, device model-specific, and device instance-specific updates. Before cut-over, the upload/download object of the device may perform safety checks on a received update to assure that an update is appropriate for a specific device type. As all the communication takes place between application objects, such an update is protected by the end to end TL security mechanism. In addition, the firmware update may use security mechanisms to authenticate the update. As part of the firmware upgrade process, the upload/download object of the device may be provided with the appropriate standardized labeling, versioning, and security information for these verifications by the host update application. These verifications may be vendor-specific and are not specified by this standard.

NOTE The UDO in the DMAP to update a firmware is described here. Details about general upload/download objects that are available for use by general application processes are given in 12.15.2.4.

6.2.7.4 Layer management objects

The set of objects within the DMAP shall include objects representing access to each of the layer management SAPs. These objects include:

- the application sublayer management object (ASLMO), which provides access to the ASMSAP;
- the TL management object (TLMO), which provides access to the TMSAP;
- the NL management object (NLMO), which provides access to the NMSAP; and
- the data-link management object (DLMO), which provides access to the DMSAP.

The services, defined by the layer SAP definitions, are reflected in the features of the management objects such that, effectively, the services made available at the management SAPs become remotely accessible using secure standard communications mechanisms. Generically, the management SAPs provide for reading and writing attributes, invoking methods, and reporting events. The various layer specifications specify the exact features that are available on each of these SAPs. In effect, these specifications define the layer management objects. See 6.2.8.2 for more details on the layer management objects.

6.2.7.5 Device security management object

The DMAP shall include a device security management object (DSMO) that provides appropriately limited access to device security management functions. The DSMO manages security key material and cryptographic operations. The details of this object are provided in 7.11.

6.2.7.6 Device provisioning object

The DMAP shall include a device provisioning object (DPO) that is accessed during the provisioning process of the device. More details about the attributes, methods and alerts of the DPO are provided in 13.9.

6.2.7.7 Health reports concentrator object

The DMAP shall include a health reports concentrator object (HRCO) that can be configured by the system manager to enable periodic publication of device health reports. The device health reports may consist of periodic publication of one or more attributes from the management objects in the DMAP.

The attributes of the HRCO are as per the definition of the concentrator object given in 12.15.2.5. The object identifier of the HRCO in the DMAP shall be 10. The CommunicationEndPoint and Array of ObjectAttributeIndexAndSize attributes of the HRCO are used by the system manager to set up periodic publications of health reports from the device. The system manager may choose to include any attribute from any management object in such health reports which are used for system performance monitoring. System performance monitoring is described in 6.3.7.

6.2.8 Functions of device management and layer management

6.2.8.1 Device management functions

6.2.8.1.1 General

Device management capabilities are provided primarily via access to DMO attributes and invocation of methods. The DMO contains critical attributes of device-wide scope that shall be available in all devices. Product implementers may extend the list of attributes beyond the required attributes described below.

Some attributes available via the DMO may also be available as an attribute of a particular layer management object. In that case, change in the value of any such attribute shall be reflected in the corresponding attribute.

The DMO shall provide system time information to other management objects; the DMO may obtain this system time information by interacting with the DL of the device and/or the system

manager or from another source, such as a GPS receiver within the device. Time-keeping by the DL is described in the 9.1.9. The role of the system manager in maintaining time across the network is described in 6.3.10.

The establishment, modification and termination of contracts for a device shall be managed by its DMO. Contracts are described in 6.3.11.2.

The attributes of the DMO are defined in Table 10.

Table 10 – DMO attributes *(1 of 8)*

Standard object type name: Device management object (DMO)				
Standard object type identifier: 127				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
EUI64	1	64-bit unique identifier of device	Type: EUI64Address Classification: Constant Accessibility: Read only Default value: 0x0000 0000 0000 0001	This shall be a global unique EUI64Address. This attribute is a duplicate of the corresponding attributes in the DLMO and NLMO
DL16Address	2	16-bit identifier for device, unique in its D-subnet	Type: DL16Address Classification: Static Accessibility: Read/write Default value: 0 Valid range: 0: address unassigned; 1..0x7FFF: unicast address	Address unique in the D-subnet of the device; assigned by the system manager. This attribute is a duplicate of the corresponding attribute in the DLMO and NLMO. Configured by the system manager during the device's subnet joining process
IPv6Address	3	IPv6Address assigned by system manager	Type: IPv6Address Classification: Static Accessibility: Read/write Default value: 0 Valid range: 0: address unassigned; other with higher-order bit reset: unicast address.	Network address unique in the network of the device and used by the application to identify the devices across the network. This attribute is a duplicate of the corresponding attributes in DLMO and NLMO. Configured by the system manager during the device's subnet joining process

Table 10 *(2 of 8)*

Standard object type name: Device management object (DMO)				
Standard object type identifier: 127				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Device_Role_Capability	4	Role(s) that the device is capable of playing in the network; roles are defined in 5.2.6.2	Type: BitArray16 Classification: Constant Accessibility: Read only	This attribute shall be sent to the system manager during the device's subnet joining process; see 6.3.9.2. See 5.2.6.2 for acceptable roles and their descriptions. Named indices: 0: I/O; 1: router; 2: backbone router; 3: gateway; 4: system manager; 5: security manager; 6: system time source; 7: provisioning device; 8..15: reserved (shall be 0)
Assigned_Device_Role	5	Role(s) of the device as assigned by the system manager; roles are defined in 5.2.6.2	Type: BitArray16 Classification: Static Accessibility: Read/write	This attribute shall be written by the system manager during the device's subnet joining process; see 6.3.9.2. Refer to 5.2.6.2 for acceptable roles and their descriptions. The assigned role for a device shall not exceed its capabilities as specified in attribute 4. The bit array indices are identical to those of attribute 4
Vendor_ID	6	Human-readable identification of device vendor	Type: VisibleString16 Classification: Constant Accessibility: Read only	Assigned by vendor during device manufacturing
Model_ID	7	Human-readable identification of device model	Type: VisibleString16 Classification: Constant Accessibility: Read only	Assigned by vendor during device manufacturing
Tag_Name	8	Tag name of device	Type: VisibleString16 Classification: Static Accessibility: Read/write	Assigned by user
Serial_Number	9	Serial number of device	Type: VisibleString16 Classification: Constant Accessibility: Read only	Assigned by vendor during device manufacturing

Table 10 *(3 of 8)*

Standard object type name: Device management object (DMO)				
Standard object type identifier: 127				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Power_Supply_Status	10	Status information of power supply of device	Type: Unsigned8 Classification: Dynamic Accessibility: Read/write	Named values: 0: line powered; 1: battery powered, greater than 75 % remaining capacity; 2: battery powered, between 25 % and 75 % remaining capacity; 3: battery powered, less than 25 % remaining capacity
Device_Power_Status_ Check_AlertDescriptor	11	Used to change the priority of Device_Power_Status_Check alert (described in Table 11); this alert can also be turned on or turned off	Type: Alert report descriptor Classification: Static Accessibility: Read/write Default value: [FALSE, 8]	—
DMAP_State	12	Status of DMAP	Type: Unsigned8 Classification: Dynamic Accessibility: Read only Default value: 1: active	DMAP state diagram is same as UAP state diagram given in 12.15.2.2.3. Named values: 0: inactive; 1: active; 2: failed

Table 10 *(4 of 8)*

Standard object type name: Device management object (DMO)				
Standard object type identifier: 127				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Join_Command	13	Command informing the device to join the system, restart itself and re-join, or reset to factory defaults	Type: Unsigned8 Classification: Static Accessibility: Read/write Default value: 0: none	The use of this attribute is described in 6.3.9. The value 0: none shall not be indicated in a write request. Only the provisioning device is expected to be able to issue the Join_Command with value 1 = join / start, since the device has not yet joined the network and so is not accessible by any other device. WarmRestart shall preserve static and constant attributes data including contracts and T-keys. RestartAsProvisioned corresponds to the provisioned state of the device in which the device only retains the information that is received during its provisioning step. Reset to factory defaults corresponds to the unconfigured device phase in Figure 5. Named values: 0: none; 1: join and start; 2: warm restart; 3: restart as provisioned; 4: reset to factory defaults
Static_Revision_Level	14	Revision level of the static data associated with all management objects	Type: Unsigned32 Classification: Dynamic Accessibility: Read only Default value: 0: none	Revision level is incremented each time a static attribute value in any management object is changed; the value rolls over when the limit is reached; the value resets whenever the device is reset to factory defaults (Join_Command value of 4: reset to factory defaults)

Table 10 *(5 of 8)*

Standard object type name: Device management object (DMO)				
Standard object type identifier: 127				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Restart_Count	15	Number of times device restarted	Type: Unsigned16 Classification: Static Accessibility: Read only Default value: 0	Device restart can be due to battery replacement, warm restart command, firmware download, link failure; the value rolls over if the max value is reached; the value resets to 0 when the device is reset to factory defaults
Uptime	16	Low accuracy counter for counting seconds since last device restart	Type: Unsigned32 Classification: Dynamic Accessibility: Read only Default value: 0	Units in seconds; reset to 0 if device restarts
Device_Memory_Total	17	Total memory of device expressed in octets	Type: Unsigned32 Classification: Constant Accessibility: Read only	Units in octets
Device_Memory_Used	18	Memory currently used in device expressed in octets	Type: Unsigned32 Classification: Dynamic Accessibility: Read only	Units in octets
TAI_Time	19	Current TAI time	Type: TAINetworkTime Classification: Dynamic Accessibility: Read only	Value is obtained either from the DL (if the device is not system time source) or from the backbone/external source (if the device is system time source or is on the backbone and does not have a DL)
Comm_SW_Major_Version	20	Major version of communications software currently being used in the device	Type: Unsigned8 Classification: Constant Accessibility: Read only Default value: 0	8-bit communications software major version number, assigned by this standard, equals 0
Comm_SW_Minor_Version	21	Minor version of communications software currently being used in the device	Type: Unsigned8 Classification: Constant Accessibility: Read only Default value: 1	8-bit communications software minor version number assigned by this standard, equals 1
Software_Revision_ Information	22	Revision information about communications software for particular major and minor version numbers	Type: VisibleString16 Classification: Constant Accessibility: Read only	Revision information assigned by vendor

Table 10 *(6 of 8)*

Standard object type name: Device management object (DMO)				
Standard object type identifier: 127				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
System_Manager_ IPv6Address	23	Network address of system manager	Type: IPv6Address Classification: Static Accessibility: Read/write Default value: 0	This information shall be provided to the device either during the provisioning process or during the device's subnet joining process
System_Manager_ EUI64Address	24	EUI64Address of system manager	Type: EUI64Address Classification: Static Accessibility: Read/write Default value: 0	This information shall be provided to the device either during the provisioning process or the joining process
System_Manager_ DL16Address	25	DL16Address of system manager in D-subnet of device	Type: DL16Address Classification: Static Accessibility: Read/write Default value: 0	This attribute shall be configured by the system manager during the joining process
Contracts_Table	26	Table that includes information about all existing contracts of the device	Type: Array of Contract_Data Classification: Static Accessibility: Read/write	Updated when a corresponding contract gets established, modified, renewed or terminated; see 6.3.11.2 for more details about contracts and about data type Contract_Data. A new entry in the Contracts_Table shall be created each time a contract response associated with the successful creation of a new contract is received from the system manager. For additional details see 6.3.11
Contract_Request_ Timeout	27	Timeout for DMO before the contract request can be retried	Type: Unsigned16 Classification: Static Accessibility: Read/write Default value: 30 s	System manager sets this timeout value after the device joins the network. Unit: s

Table 10 *(7 of 8)*

Standard object type name: Device management object (DMO)				
Standard object type identifier: 127				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Max_ClientServer_Retries	28	The maximum number of client request retries DMAP shall send in order to have a successful client/server communication	Type: Unsigned8 Classification: Static Accessibility: Read/write Default value: 3 Valid range: 0..8	The number of retries sent for a particular message may vary by message based on the application process determination of the importance of the message. For example, some messages may not be retried at all, and others may be retried the maximum number of times
Max_Retry_Timeout_ Interval	29	The maximum timeout interval for a client request before it is sent again	Type: Unsigned16 Classification: Static Accessibility: Read/write Default value: 30 s	System manager sets this timeout value after the device joins the network. Unit: s
DMAP_Objects_Count	30	Number of management objects in DMAP including this DMO	Type: Unsigned8 Classification: Static Accessibility: Read only Default value: 1 Valid range: > 0	Total count of the management objects such as DLMO, NLMO, etc., in the DMAP of this device; all application processes in the device shall include an attribute with such information
DMAP_Objects_List	31	List of all the management objects in the DMAP	Type: Array of ObjectIDandType Classification: Static Accessibility: Read only	List to identify all the management objects that are available in the DMAP; all application processes in the device shall include an attribute with such information. See 12.16.3.10 for details about this data type
Metadata_Contracts_Table	32	Metadata (count and capacity) of the Contracts_Table attribute	Type: Metadata_attribute	Metadata containing a count of the number of entries in the table and capacity (the total number of rows allowed) for the table; see 6.2.6.3 for details about this data type

Table 10 *(8 of 8)*

Standard object type name: Device management object (DMO)				
Standard object type identifier: 127				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Non_Volatile_Memory_ Capability	33	Indicates if device is capable of maintaining all DMAP information that falls under the Static classification in non-volatile memory over a power-cycle or not	Type: Boolean8 Classification: Constant Accessibility: Read only	See 6.3.9.4.2 for more information
Warm_Restart_Attempts_ Timeout	34	The timeout after which a device that is trying to re-join the network through a warmRestart converts to a restartAsProvisioned command	Type: Unsigned16 Classification: Static Accessibility: Read/write Default value: 60	Units in minutes; see 6.3.9.4.2 for more information
Device_Restart_ AlertDescriptor	35	Used to change the priority of Device_Restart alert (described in Table 11); this alert can also be turned on or turned off	Type: Alert report descriptor Classification: Static Accessibility: Read/write Default value: [FALSE, 8]	—
Proxy_Join_Request_Rate	36	Used to control the maximum rate at which a proxy router will accommodate join requests	Type: Integer8 Classification: Static Accessibility: Read/write Default value: 6 Valid range: -4..127	Minimum required interval between join requests that the proxy router is permitted to accept. A value of $N > 0$ specifies a period of N s, while $N < 0$ specifies a period of -1/N s. $N = 0$ disables the attribute. This parameter is used to reduce the impact of denial of service (DoS) attacks.
Reserved for future editions of this standard	37..63	—	—	—

6.2.8.1.2 Device management object alerts

The DMO of the device shall send an alert to indicate a change in its power status. The DMO of the device shall send an alert whenever it goes through a device restart. Device restart is described in 6.3.9.4.2.

The alerts of the DMO are defined in Table 11.

Table 11 – DMO alerts

Standard object type name(s): Device management object (DMO)					
Standard object type identifier: 127					
Description of the alert: Communication diagnostic alerts to indicate that the device power supply status has changed and to indicate that the device has restarted					
Alert class (Enumerated: alarm or event)	**Alert category (Enumerated: device diagnostic, comm. diagnostic, security, or process)**	**Alert type (Enumerated: based on alert category)**	**Alert priority (Enumerated: urgent, high, med, low, journal)**	**Value data type**	**Description of value included with alert**
0 = Event	1 = Comm. diagnostics	0 = Device_Power_Status_Check	8 = Medium	Type: Unsigned8	The current value of the Power_Supply_Status attribute in Table 10 is included in this alert
0 = Event	1 = Comm. diagnostics	1 = Device_Restart	8 = Medium	Type: N/A	Only a device that has DMO attribute Non_Volatile_Memory_Capability = 1 can send this alert to indicate that it has gone through a warm restart

6.2.8.1.3 Device management object methods

The methods of the DMO are described in 6.3.9.2.2, 6.3.11.2.10.5, and 6.3.11.2.11.3.

6.2.8.2 Layer management

6.2.8.2.1 General

Each communication layer within the protocol suite has a self-contained layer management functionality. Each layer management function provides a management SAP. Device management of layers is accomplished via access to the management SAPs on each of the layers, as shown in Figure 25. Management of the layers within a device may be done locally by a functionality that resides within the DMO, since the DMO has access to the management SAPs. In addition, as discussed in 6.2.4, layer management may be accomplished remotely by a system manager.

The formal definition of each of the layer management objects is included below. The definition of the layer management objects within the DMAP corresponds directly with the layer management SAP definition. However, there may be a need to restrict remote access to specific features of a given layer management SAP. Thus, some attributes or methods, while accessible to the local DMO, should not be remotely accessible. Such restrictions, whenever necessary, are specified in the layer specifications. The operations described in Annex J can be used to access attributes in these layer management objects.

6.2.8.2.2 DL management object

6.2.8.2.2.1 General

In the architecture defined by this standard, all DL, MAC, and PhL layer management services are provided via a unified data-link management service access point (DMSAP). There are no

directly accessible management SAPs for the physical layer and MAC layer, because those layers sometimes require management actions to be time-synchronous with data flow. Thus the DLMO provides the attributes of those layers that are accessible remotely.

6.2.8.2.2.2 Physical layer management

Physical layer management entities are manipulated indirectly via the DMSAP. DL attributes relate to a subset of those defined in IEEE 802.15.4, as described in 9.1.5.

6.2.8.2.2.3 Media access control sublayer management

MAC sublayer management entities are manipulated indirectly via the DMSAP. DL attributes relate to a subset of those defined in IEEE 802.15.4, as described in 9.1.5.

6.2.8.2.2.4 DL management

DL management attributes and methods are available via the DMSAP. Attributes, methods, and alerts of the DLMO are defined in 9.4 and 9.6.

6.2.8.2.2.5 NL management

NL management attributes and methods are available via the NMSAP. Attributes, methods and alerts of the NLMO are defined in 10.4.

6.2.8.2.2.6 TL management

TL management attributes and methods are available via the TMSAP. Attributes, methods, and alerts of the TLMO are defined in 11.6.

6.2.8.2.2.7 Security management

Device security management attributes and methods are available via the DMSAP and TMSAP. Attributes, methods and alerts of the DSMO are defined in 7.11.

6.2.8.2.2.8 Application sublayer management

Application sublayer management attributes and methods are available via the ASMSAP. Attributes, methods, and alerts of the ASLMO are defined in 12.19.

6.3 System manager

6.3.1 General

The functions of the system manager include security management, address allocation, software updating, system performance monitoring, device management, system time services, and communication configuration including contract services, and redundancy management.

The system manager shall use ASL services to remotely access management objects in the DMAPs of devices compliant with this standard.

System manager is a role and is not tied into a specific fixed physical address.

6.3.2 System management architecture

Conceptually, the system manager can be viewed as an application process running on any device in the network. Such a device shall be capable of supporting the system manager role. The SMAP is accessible only on such a device. The SMAP shall use the application sublayer SAP ASLDE-1 SAP for communicating with the devices. This application sublayer SAP shall correspond to TL SAP TDSAP-1 which shall correspond to port number 0xF0B1.

Figure 26 shows the system manager that resides in a field device compliant with this standard.

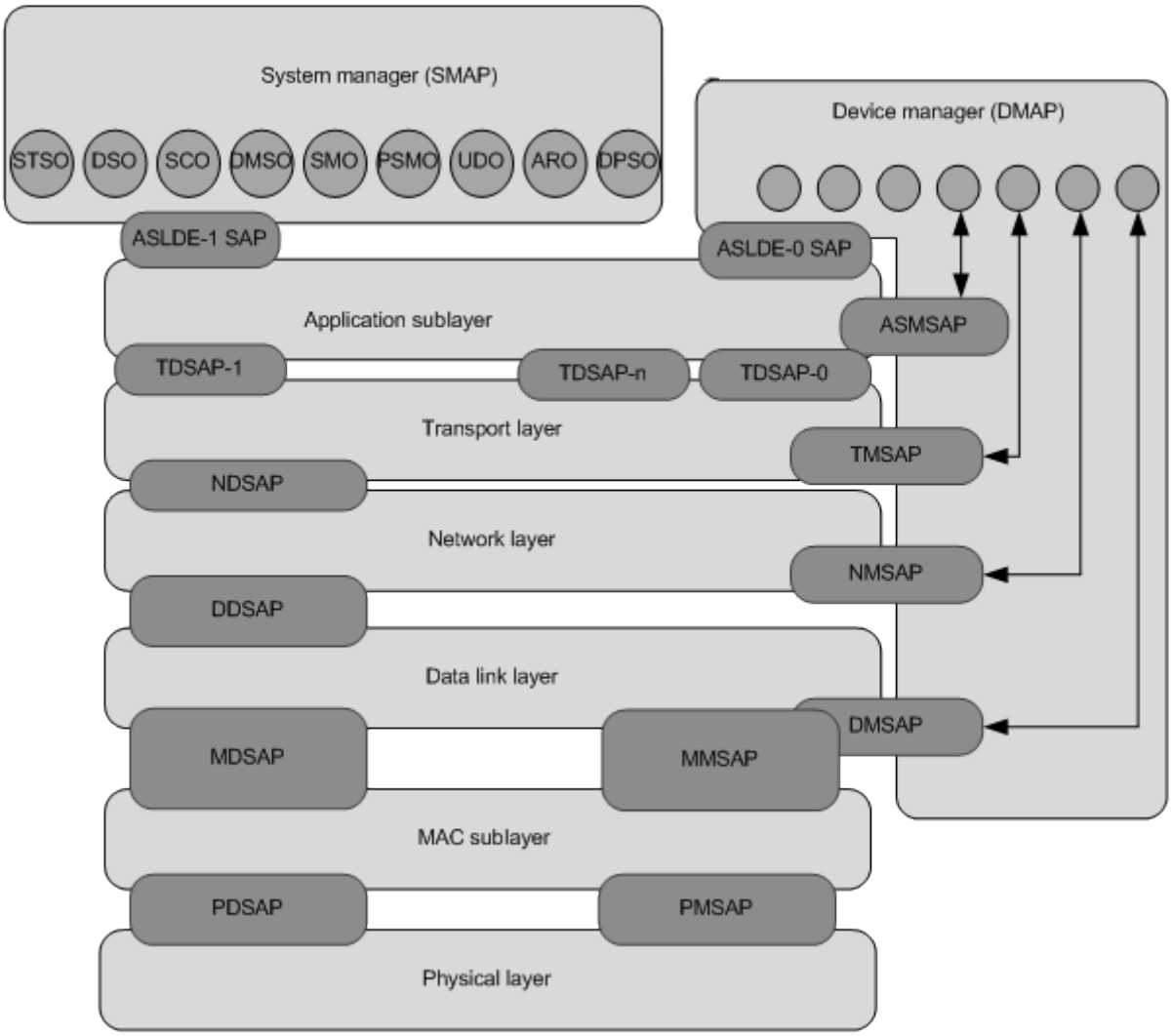


Figure 26 – System manager architecture concept

As shown in Figure 26, TDSAP-1 shall be used to access the management objects in the SMAP. The definition of these system management objects is necessary to provide remote access to these functions for the devices in the network that are compliant with this standard.

6.3.3 Standard system management object types

Table 12 includes a list of system management object types that are specified in this standard.

Table 12 – System management object types

Standard object type name	Standard object type ID	Standard object identifier	Object description
System time service object (STSO)	100	1	This object facilitates the management of system-wide time information; see 6.3.10
Directory service object (DSO)	101	2	This object facilitates the management of addresses for all existing devices in the network; see 6.3.5
System communication configuration object (SCO)	102	3	This object facilitates the communication configuration of the system including contract establishment, modification and termination; see 6.3.11
Device management service object (DMSO)	103	4	This object facilitates device joining, device leaving, and device communication configuration; see 6.3.9
System monitoring object (SMO)	104	5	This object facilitates the monitoring of system performance; see 6.3.7
Proxy security management object (PSMO)	105	6	This object acts as a proxy for the security manager; see 6.3.4
Upload/download object (UDO)	3	7	This object facilitates downloading firmware/data to devices and uploading data from devices; see 6.3.6
Alert-receiving object (ARO)	2	8	This object receives all the alerts destined for the system manager; see 6.3.7
Device provisioning object (DPO)	106	9	This object facilitates device provisioning; see 6.3.8
Reserved for future editions of this standard	107..113	—	—

Devices that require system management services communicate with the appropriate objects given above.

6.3.4 Security management

The system manager interfaces with the security manager to generate keys and authenticate devices. The security manager is functionally separated from the system manager so that the security policies can be common across the networks of the administrator and other types of networks. Placing the security manager functionally behind the system manager also hides from the devices the various protocols, such as Kerberos, that may be used by a security management function. More details are provided in Clause 7.

The interface between the system manager and the security manager is not specified in this standard. Conceptually, the system manager can be viewed as including a proxy security management object (PSMO). This PSMO forwards all security related messages between the security manager and the devices in the network that are compliant with this standard. This PSMO can be used by the security manager to access information from other system management objects, such as current TAI time, if necessary. The security manager does not have a valid address as defined by this standard; thus, devices that wish to communicate with the security manager can only do so by communicating with the PSMO.

The attributes, methods and alerts of the PSMO are defined in Clause 7.

6.3.5 Addresses and address allocation

6.3.5.1 General

The system manager is responsible for assigning addresses to devices when they join the network.

6.3.5.2 Address types

Every device compliant with this standard shall have one identifier and two addresses, as follows.

- Each device compliant with this standard shall have an EUI64Address identifier that is presumed to be globally unique (vendors are expected to ensure global uniqueness of these identifiers). Failover mechanisms for the gateway, system manager and security manager roles are usually provided through redundancy. Redundancy for the purposes of failover may involve EUI64Address identifier duplication for the redundant entities. Such EUI64Address identifier duplication is outside the scope of this standard.
- Each device compliant with this standard shall be assigned one IPv6Address by the system manager when it joins the network; this IPv6Address shall be unique across the network.
- Each device compliant with this standard that is accessible through a D-subnet shall have a D-subnet-unique DL16Address for its IPv6Address. This DL16Address shall be assigned by the system manager. The scope of any DL16Address is limited to a particular D-subnet.

The ranges and uses of 16-bit D-addresses are:

– 0x0000: reserved by this standard to indicate that a DL16Address has not been assigned to the device;

– 0x0001..0x7FFF: reserved by IETF RFC 4944 for 6LoWPAN unicast device addressing;

– 0x8000..0xBFFF: reserved by IETF RFC 4944 for 6LoWPAN multicast;

– 0xC000..0xCFFF: reserved by this standard for graph IDs;

– 0xD000..0xFFFD: reserved;

– 0xFFFE: reserved by IEEE 802.15.4e for the local PAN coordinator; and

– 0xFFFF: reserved by IEEE 802.15.4 for local broadcast.

In this standard, DL16Addressing is always used within a D-subnet, with the exception that EUI64Address identifiers are used in a limited way during the joining process until a joining device has received a subnet-local DL16Address from the system manager. The joining process is described in 7.4.

6.3.5.3 Address allocation

The system manager shall allocate the IPv6Address, as well as the D-subnet-unique DL16Address, to a device when it joins the network. This is described in 7.4.

When a source device that belongs to a particular D-subnet communicates over the backbone with a destination device that belongs to a different D-subnet, the system manager shall assign local D-subnet-unique DL16Addresses to both devices in each other's D-subnets. Such local DL16Addresses for remote devices (i.e., devices residing in another D-subnet) may be established by the system manager upon a contract request. (Contracts are described in 6.3.11.2.)

When the source sends a message to the destination, the DL16Address of the destination shall be used at the NL and DL to construct the NPDU and DPDU. These layers can use the directory look-up service provided by the system manager to obtain the DL16Address of the destination, if not already known. This service is described in 6.3.5.4.

The backbone routers shall do the address translations between the DL16Address (per D-subnet) and IPv6Address for a given device. Note that the DL16Address is used only within the DL. Once a message reaches the backbone, the full IPv6Address shall be used. Informative examples for such scenarios are given in 10.2.7.

This standard does not specify any mechanisms for how the system manager allocates the IPv6Addresses and DL16Addresses. Subclause 10.2.7 contains examples for Ethernet based routing and fieldbus based routing. The Ethernet-based routing example describes the use of IPv6-based IPv6Addresses. The fieldbus routing example describes the use of non-IPv6-based IPv6Addresses.

Devices shall only have one valid IPv6Address. Multi-homing devices that require multiple IPv6Addresses are not covered in this standard.

Addressed entities on the backbone, such as system managers and gateways, shall also be assigned DL16Addresses for use within a D-subnet. Thus, the addresses most used within a D-subnet will be DL16Addresses.

6.3.5.4 Directory service

The directory service object (DSO) in the system manager provides the necessary attributes for looking up the address translation between the EUI64Address, the IPv6Address, and the DL16Address(es) of a given device. D-subnet IDs are also maintained by the DSO, but the allocation of these D-subnet IDs is not specified in this standard.

The attributes of the DSO are defined in Table 13.

Table 13 – DSO attributes

Standard object type name: Directory service object (DSO)				
Standard object type identifier: 101				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Address_Translation_Table	1	Address translation table containing EUI64Address, IPv6Address, D-subnet ID(s) and DL16Address(es) of all devices in the network	Type: Array of Address_Translation_Row Classification: Dynamic Accessibility: Read only	Structured attribute used to look-up address translations; see Table 14
Reserved for future editions of this standard	2..63	—	—	—

The data structure Address_Translation_Row is defined in Table 14.

Table 14 – Address_Translation_Row data structure

Standard data type name: Address_Translation_Row		
Standard data type code: 402		
Element name	**Element identifier**	**Element type**
EUI64Address	1	Globally unique EUI64Address of device; Type: EUI64Address Classification: Static Accessibility: Read only
IPv6Address	2	IPv6Address of device assigned by system manager; Type: IPv6Address Classification: Static Accessibility: Read only
DL_Subnet_ID	3	D-subnet in which or from which this device is reachable; a device may be reachable from multiple D-subnets in which case this element corresponds to one such D-subnet; Type: Unsigned16 Classification: Static Accessibility: Read only
DL16Address	4	DL16Address of device in the D-subnet indicated by the DL_Subnet_ID element given above; Type: Unsigned16 Classification: Static Accessibility: Read only

Address translation look-up service is provided by the Read_Address_Row method defined in Table 15.

Table 15 – Read_Address_Row method

Standard object type name: Directory service object (DSO)				
Standard object type identifier: 101				
Method name	**Method ID**	**Method description**		
Read_Address_Row	1	Method to use the address translation look-up service for reading the values of other addresses/identifiers of a device given an index (one of its address/identifier)		
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Attribute_ID	Data type: Unsigned8	Value = 1 (Address_Translation_Table attribute of DSO)

Standard object type name: Directory service object (DSO)				
Standard object type identifier: 101				
	2	Index_Info	Data type: Unsigned8	Named indices: 0: only EUI64Address provided; 1: only IPv6Address is provided; 2: EUI64Address and D-subnet ID are provided; 3: IPv6Address and D-subnet ID are provided; 4: D-subnet ID and DL16Address are provided. See Table 16
	3	Index_EUI64	Data type: EUI64Address	Value: EUI64Address of device for which address look-up is needed
	4	Index_128_Bit_Address	Data type: IPv6Address	Value: IPv6Address of device for which address look-up is needed
	5	Index_DL_Subnet_ID	Data type: Unsigned16	Value: D-subnet ID of device for which address look-up is needed
	6	Index_DL_address_16_Bit	Data type: DL16Address	Value: DL16Address of device for which address look-up is needed

	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Value_Type	Data type: Unsigned8	Indicates the type of information being provided. Named values: 0: single row if D-subnet ID value provided as input argument; 1: all rows if D-subnet ID value not provided as input argument (i.e., all rows for given EUI64Address or IPv6Address are returned). See Table 17
	2	Value_Size	Data type: Unsigned8	Number of rows being returned
	3	Data_Value_1	Data type: Address_Translation_Row	EUI64Address, IPv6Address, D-subnet ID, DL16Address of device
	2+*n*	Data_Value_*n*	Data type: Address_Translation_Row	EUI64Address, IPv6Address, D-subnet ID, DL16Address of device

Some of the input arguments are not applicable if the Index_Info argument has certain values and so shall not be included in the request. The usage of input arguments for the Read_Address_Row method is described in Table 16.

Table 16 – Input argument usage for Read_Address_Row method

Input argument	Not applicable for Index_Info value
Attribute_ID	—
Index_Info	—
Index_EUI64	1, 3, 4
Index_128_Bit_Address	0, 2, 4
Index_DL_Subnet_ID	0, 1
Index_DL_Address_16_Bit	0, 1, 2, 3

Some of the output arguments are not applicable if the Value_Type argument has certain values and so shall not be included in the response. The usage of output arguments for the Read_Address_Row method is described in Table 17.

Table 17 – Output argument usage for Read_Address_Row method

Output argument	Not applicable for Value_Type value
Value_Type	—
Value_Size	0
Data_Value_1	—
Data_Value_n	0

High-side interfaces on the DSO to delete addresses or modify addresses are not specified in this standard.

6.3.5.5 Multicast DL16Address management

DL16Addresses of the form 0x 100x xxxx xxxx xxxx shall be reserved for multicast, following the convention set by IETF RFC 4944.

Multicast DL16Address management is not specified in this standard. Additional attributes for the directory service object may be specified by vendors that support multicast DL16Address management.

6.3.6 Firmware upgrade

The system manager provides support for over-the-air firmware upgrades to devices. The system manager supports communication protocol suite firmware updates.

The system manager shall provide an interface for accepting the firmware upgrades that need to be sent to any device in the network. The system manager shall use the UDO in the DMAP of the device for sending this update to the device. Communication protocol suite firmware updates shall be performed only through the system manager UDO. This UDO is described in 12.15.2.4.

As the system manager maintains information about all the devices in the network, the host update application can obtain information about the devices that are in the network from the system manager. The host update application may use this information to determine which devices need such firmware upgrades. The gateway may also be used to send firmware upgrades to the device. If these upgrades are communication protocol suite upgrades, they shall be sent through the system manager UDO. This is described in U.3.2.

The firmware upgrade process shall assure that network operations are maintained across updates. This process may include a cut-over mechanism that specifies a cut-over time (after the update is delivered), at which point devices shall begin using the new firmware. The cut-over time shall use the shared sense of time configured by the system manager. If the system manager is sending the firmware upgrade to the device, it may send the cut-over time along with the download, or it may send the cut-over after the download is complete.

Since firmware upgrades may be vendor-specific, the updates are opaque to the system manager providing the update service. The system manager accepts updates via unspecified protocols (e.g., a user interface tool with a DVD reader) and provides updating to selected devices at specified times based on user or other input. Details of how the host update application communicates with the system manager, such as what devices should be updated, what their vendor IDs are, when the devices should be updated, and in what order they should be updated are not specified by this standard, but such functions are expected to be supported by the system manager. The system manager may schedule updates to the devices in such a manner that network downtime is either avoided or minimized. This schedule may depend on network topology.

Multicasting of firmware upgrades is not specified by this standard.

The UDO in the system manager shall be used if the firmware of the system manager itself needs to be upgraded. The attributes, methods and state machines of the UDO in the system manager are as per the definition provided in 12.15.2.4. The object identifier for the UDO in the SMAP shall be 7.

6.3.7 System performance monitoring

6.3.7.1 General

System performance monitoring is done by the system manager in order to collect information that can be used to take necessary actions for optimizing system performance and for reacting to changes in the radio environment and device status. Such actions are done through system communication configuration which is described in 6.3.11.

System performance monitoring is accomplished via polling of device attributes or by configuring devices to generate alerts that provide event-driven information.

System performance monitoring using periodic publication of health reports from the devices is supported through the use of the HRCO in the DMAP of each device. HRCO is described in 6.2.7.7 and can be configured by the system manager to periodically report the values of one or more attributes in the management objects of the device. Before the system manager configures the HRCO of any particular device to publish health reports, it needs to create a unique dispersion object in the SMAP to act as the subscriber to the data that will be published by the HRCO of this particular device. Information about this unique dispersion object shall be conveyed to the HRCO of this particular device by configuring the CommunicationEndPoint attribute in the HRCO. The dispersion object is described in 12.15.2.6.

Information about the capabilities of a new device is provided to the system manager during its joining process. This is discussed in 6.3.9.

Devices may be configured by the system manager to generate alerts to provide event-driven information, for example, when a link stops working or when the battery of a field device has less than 25 % remaining capacity. This is described in 6.3.7.2.

While the device implementing the system manager may have an interface that allows plant operations and maintenance personnel to observe and control the performance of the network and devices, this interface is neither mandatory nor is it specified by this standard.

The UDO in the DMAP of a device may be used for downloading large blocks of device performance data from the device to the system manager. Such data may be vendor-specific, device model-specific, or device instance-specific. The UDO in the system manager may be used to upload such data for further analysis. Such data collection and analysis is not specified by this standard.

6.3.7.2 System management alerts

The system manager contains an alert-receiving object (ARO) that receives communication diagnostics alerts and security alerts. Such alerts are described in 6.2.7.2. These alerts may be used by the system manager to monitor system performance and take suitable action when necessary. The object identifier for the ARO in the SMAP shall be 8.

After a device joins the network, it shall set the Alert_Master_Comm_Diagnostics and Alert_Master_Security attributes of the ARMO to point to the system manager.

The attributes of the ARO in the system manager are as per the definition given in 12.15.2.3. The default value for the ARO.Categories attribute shall be 0110 0000.

The state diagram describing the handling of alert reports by the ARO is given in 12.15.2.3.

6.3.7.3 System monitoring object

The attributes of the SMO are given in Table 18.

Table 18 – Attributes of SMO in system manager

Standard object type name: System monitoring object (SMO)				
Standard object type identifier: 104				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Reserved for future editions of this standard	1..63	—	—	—

6.3.7.4 System monitoring configuration

System performance monitoring may have its own dynamic configuration such that monitoring may be increased when necessary. For example, the list of active alerts may be adjusted depending on the current state of the network. For network diagnostics, if a failure mode is suspected, activating specific alerts may provide evidence of that failure. Such configuration of the system manager is not specified by this standard.

6.3.8 Device provisioning service

Before a device joins the network, it requires the appropriate security credentials and network-specific information. These are provided to the device during the provisioning process. The provisioning process, the role of the system manager in this process, and the definition of the device provisioning object (DPO) are described in Clause 13.

6.3.9 Device management services

6.3.9.1 General

A device compliant with this standard may go through several phases in its operational lifetime. These phases are described in 5.2.8. The management of a device through some of these phases is performed by the system manager. Specifically, the system manager plays a role in the joining and leaving processes as well as the communication configuration of a device.

6.3.9.2 Joining process

6.3.9.2.1 General

A new device shall obtain the necessary provisioning information from the provisioning device. This is described in Clause 13. The Join_Command attribute in the DMO of a device shall be used to command the device to join the network. Only the provisioning device can set the Join_Command attribute to 1, hence explicitly triggering the joining process of the device, given that the device has no connectivity with other entities in the network. The Join_Command attribute shall be set to 1 by the provisioning device only following a successful provisioning of the device. Advertising routers shall proxy join requests at the rate indicated by the DMO attribute Proxy_Join_Request_Rate (attribute 36). This rate ensures that the network is protected from denial of service attacks attempted via the proxy routers. For a description of Proxy_Join_Request_Rate, see Table 10.

The system manager controls the process of new devices joining the network. Non-joined devices that implement a DLE as per this standard listen for advertisement messages from local routers whose advertisement functions are configured by the system manager. Such an advertising router shall assist during the joining process of a new device by acting as a proxy for the system manager. This advertising router forwards the join request from the new device to the system manager and forwards the join response from the system manager to the new device. A join request from the new device shall be processed by the system manager. The system manager shall generate a join response after communicating with the security manager. Advertising routers shall proxy join requests at the rate indicated by the DMO attribute Proxy_Join_Request_Rate (attribute 36). This rate protects the network from denial of service attacks attempted by the proxy routers. For a description of this attribute, see Table 10.

The join request from a new device shall include non-security information such as the EUI64Address and capabilities of the device as well as security information of the device. The join response from the system manager shall include non-security information such as the assigned IPv6Address, assigned DL16Address and contract information of the device as well as security information such as T-key. The security information in the join request and join response is described in 7.4. Contracts are described in 6.3.11.2.

More details about the joining process are described in 7.4.

6.3.9.2.2 Device management object methods for advertising router

The new device shall use the Proxy_System_Manager_Join method and Proxy_System_Manager_Contract method defined for the DMO of the advertising router to send its non-security information that is part of the join request and to get its non-security information that is part of the join response. The non-security information that is part of the join request is split into the network join request and the contract request. The non-security information that is part of the join response is split into the network join response and the contract response. Contracts are described in 6.3.11.2.

The Proxy_System_Manager_Join method is defined in Table 19. The Proxy_System_Manager_Contract method is defined in Table 20.

The new device shall use the methods defined in 7.4 for the DMO of the advertising router to send its security information that is part of the join request and to get its security information that is part of the join response. The use of all these methods by the new device is described in 7.4.

Access to the DMAP of the device is restricted to the SMAP present in the system manager. The joining shall be only allowed to access the Proxy_System_Manager_Join and Proxy_System_Manager_Contract DMO methods during the joining process.

Table 19 – Proxy_System_Manager_Join method

Standard object type name: Device management object (DMO)				
Standard object type identifier: 127				
Method name	**Method ID**	**Method description**		
Proxy_System_Manager_Join	3	Method to use advertising router as proxy system manager to send network join request of a new device and get network join response		
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	EUI64	EUI64Address	DMO attribute EUI64; see Table 10
	2	DL_Subnet_ID	Unsigned16	D-subnet that the new device is trying to join, which is also the D-subnet of the advertising router. Data value: 0: device is not part of any D-subnet
	3	Device_Role_Capability	Unsigned16	DMO attribute Device_Role_Capability; see Table 10
	4	Size_of_Tag_Name	Type: Unsigned8	Size in octets of the Tag_Name
	5	Tag_Name	Type: VisibleString SIZE(0..16)	DMO attribute Tag_Name; see Table 10
	6	Comm_SW_Major_Version	Type: Unsigned8	DMO attribute Comm_SW_Major_Version; see Table 10
	7	Comm_SW_Minor_Version	Type: Unsigned8	DMO attribute Comm_SW_Minor_Version; see Table 10
	8	Size_of_Software_Revision_Information	Type: Unsigned8	Size in octets of the Software_Revision_Information
	9	Software_Revision_Information	Type: VisibleString SIZE(0..16)	DMO attribute Software_Revision_Information; see Table 10
	10	DeviceCapability	Type: OctetString	DLMO attribute DeviceCapability; see Table 141

Standard object type name: Device management object (DMO)				
Standard object type identifier: 127				
Method name	**Method ID**	**Method description**		
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Assigned_Network_ Address_128_Bit	Type: IPv6Address	This value is written to DMO attribute Network_Address_128_Bit; see Table 10
	2	Assigned_DL_ Address_16_Bit	Type: DL16Address	This value is written to DMO attribute DL_Address_16_Bit; see Table 10
	3	Assigned_Device_Rol e	Type: BitArray16	This value is written to DMO attribute Assigned_Device_Role; see Table 10
	4	System_Manager_ Network_Address_ 128_Bit	Type: IPv6Address	This value is written to DMO attribute System_Manager_128_Bit_Address; see Table 10
	5	System_Manager_DL _ Address_16_Bit	Type: DL16Address	This value is written to DMO attribute System_Manager_DL_Address_16_ Bit; see Table 10
	6	System_Manager_ EUI64	Type: EUI64Address	This value is written to DMO attribute System_Manager_EUI64Address; see Table 10
	7	MIC	Type: OctetString4	This value is used for protecting argument 1 through 6 with Join key. This MIC value is generated by the Security Manager. The Advertisement router shall not overwrite this value. See 7.4.4.3.2
	8	Assigned_Max_TSDU _Size	Type: Unsigned16	Indicates the maximum TSDU supported in octets which can be converted by the source into max APDU size by taking into account the TL, security, AL headers and TMIC sizes

Table 20 – Proxy_System_Manager_Contract method

Standard object type name: Device management object (DMO)				
Standard object type identifier: 127				
Method name	**Method ID**	**Method description**		
Proxy_System_ Manager_Contract	4	Method to use advertising router as proxy system manager to send contract request of a new device and get contract response. Contracts are described in 6.3.11.2		
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	EUI64	Type: EUI64Address	DMO attribute EUI64; see Table 10
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Contract_Response	Type: New_Device_Contract_Response (see Table 31	Contract response to support future communication from new device to system manager; contracts are described in 6.3.11.2
	2	MIC	Type: OctetString4	This value is used for protecting argument 1 with join key. This MIC value is generated in the Security Manager. Advertisement router shall not overwrite this value

6.3.9.2.3 Capabilities of new device

Information about the capabilities of a new device with respect to the device role shall be provided to the system manager during the joining process of the device. This information is described in Table 19.

6.3.9.3 Device configuration

A device is configured after joining a network. Device configuration includes obtaining communication resources to support the communication needs of the device and configuring the protocol stack of the device to use these resources for communication. During this configuration, the system manager may take into account the capabilities of a device. A device may be reconfigured as the network changes or as the applications on the device need to change their services.

Device configuration is usually performed during the establishment of contracts. This is described in 6.3.11.2. Attributes and methods defined for the management objects of the DMAP shall be used by the system manager to configure the device.

The system manager does not configure the UAPs on the device. This is done by host applications on plant networks or by handheld maintenance tools.

6.3.9.4 Leave process

6.3.9.4.1 General

The system manager controls the process of a previously joined device leaving the network. This leave process may be initiated by the device when it intends to leave the network, or it may be initiated by the system manager.

The leave process includes two scenarios: device restart and device reset to factory defaults.

Device restart occurs when either the device itself or the system manager causes the device to restart. Some examples that lead to this scenario are battery replacement or rebooting to apply a new firmware image. A device is reset to its factory default settings if the device is being returned to its factory default state. Some examples that lead to this scenario are the device being returned to general stock for future deployment or the device being moved to a different network.

6.3.9.4.2 Device restart

A device restart process may be initiated by the device itself or by the system manager. There are two types of restarts: warmRestart and restartAsProvisioned. In both cases, the devices will immediately initiate the joining process following the restart event. An explicit writing of the Join_Command DMO attribute to 1 is not needed following a warmRestart or a restartAsProvisioned event.

The Join_Command attribute in the DMO of a device shall be used to command the device to perform either a warmRestart or restartAsProvisioned.

A device that receives the warmRestart command shall reboot itself. If the Non_Volatile_Memory_Capability attribute in the DMO is 1, the device shall retain the values of all constant and static attributes in all application objects present in the DMAP as well as in the UAPs of the device. All other attributes are reset to their default values. If the Non_Volatile_Memory_Capability attribute in the DMO is 0, the device shall retain all the information that was provided to it during the provisioning step before it first joined the network as well as all the constant and static information present in the UAPs. All other attributes are reset to their default values and the device goes back to its provisioned state.

A device that receives the restartAsProvisioned command shall reset all constant and static attributes in all application objects present in the DMAP regardless of the Non_Volatile_Memory_Capability attribute setting present in the DMO except the DPO. A device that receives the restartAsProvisioned command shall reboot itself while retaining all the information that was provided to it during the provisioning step before it first joined the network. This information is described in Clause 13. This information is usually necessary for the device to rejoin the network without having to go through the provisioning step once again. The device shall also retain all the constant and static information present in the UAPs.

Table 21 collects and presents the effects of the different join commands on various attribute sets.

Table 21 – Effect of different join commands on attribute sets

Join Command Type	DMAP attributes (except DPO)	UAP Attributes	DPO Attributes
WarmRestart (Join_Command =2) when Non_Volatile_Memory_Capability = 1	KEEP	KEEP	KEEP
WarmRestart (Join_Command =2) when Non_Volatile_Memory_Capability = 0	CLEAR	KEEP	KEEP
RestartAsProvisioned (Join_Command = 3)	CLEAR	KEEP	KEEP
Reset to factory defaults (Join_Command = 4)	CLEAR	CLEAR	CLEAR

A firmware download may also result in a device restart. Information necessary for the device to use the new firmware and join the network may be stored in the device. The device usually goes through a restartAsProvisioned cycle in such cases.

6.3.9.4.3 Device reset to factory defaults

A device reset process may be initiated by the device itself or by the system manager.

The Join_Command attribute in the DMO of a device shall be used to command the device to perform a reset. A reset command forces the device to reset to factory defaults, and all attributes are reset to their default values. The device is expected to return to the factory default state which is described in Clause 13.

6.3.9.4.4 Device replacement

If an old device (i.e., joined device) is replaced by a new device (i.e., non-joined device) the system manager is expected to provide the old IPv6Address to this replacement device. The host application or the user is expected to inform the system manager about this replacement through communication path 5 in Figure 23. The system manager may choose to configure other attributes in the DMAP of the replacement device to match those in the old device. Configuration of the UAPs in the replacement device is expected to be done by host applications on plant networks or by handheld maintenance tools.

6.3.9.5 Device management service object

The device management service object (DMSO) in the system manager shall handle the non-security information in the join request from the new device that is forwarded by the advertising router and shall generate the non-security information in the join response.

The attributes of the DMSO are given in Table 22.

Table 22 – Attributes of DMSO in the system manager

Standard object type name: Device management service object (DMSO)				
Standard object type identifier: 103				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Reserved for future editions of this standard	1..63	—	—	—

A new device communicates with an advertising router which acts as a proxy for the system manager and forwards all the join messages between the new device and the system manager. The joining process is described in 7.4. The methods used for sending join request and join response messages between the new device and the advertising router are given in 6.3.9.2.2. The advertising router shall use the System_Manager_Join method and the

System_Manager_Contract method defined in the DMSO for sending the network join request and the contract request and for receiving the network join response and the contract response associated with the joining process of this new device.

The source object of the System_Manager_Join and System_Manager_Contract methods is the DMO of the proxy advertising router that communicates with the system manager on behalf of the new device.

The System_Manager_Join method is defined in Table 23. The System_Manager_Contract method is defined in Table 24.

Table 23 – System_Manager_Join method

Standard object type name: Device management service object (DMSO)				
Standard object type identifier: 103				
Method name	**Method ID**	**Method description**		
System_Manager_Join	1	Method to send network join request of a new device to system manager and get network join response		
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	EUI64	Type: EUI64Address	DMO attribute EUI64; see Table 10
	2	DL_Subnet_ID	Type: Unsigned16	D-subnet that the new device is trying to join; this is also the D-subnet of the advertising router. Named values: 0: device is not part of any D-subnet
	3	Device_Role_Capability	Type: Unsigned16	DMO attribute Device_Role_Capability; see Table 10
	4	Size_of_Tag_Name	Type: Unsigned8	Size in octets of the Tag_Name
	5	Tag_Name	Type: VisibleString SIZE(0..16)	DMO attribute Tag_Name; see Table 10
	6	Comm_SW_Major_Version	Type: Unsigned8	DMO attribute Comm_ SW_Major_Version; see Table 10
	7	Comm_SW_Minor_Version	Type: Unsigned8	DMO attribute Comm_ SW_Minor_Version; see Table 10
	8	Size_of_Software_ Revision_Information	Type: Unsigned8	Size in octets of the Software_Revision_ Information
	9	Software_Revision_Information	Type: VisibleString SIZE(0..16)	DMO attribute Software_Revision_ Information; see Table 10
	10	DeviceCapability	Type: OctetString	DLMO attribute DeviceCapability; see Table 141

Standard object type name: Device management service object (DMSO)				
Standard object type identifier: 103				
Method name	**Method ID**	**Method description**		
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Assigned_Network_ Address_128_Bit	Type: IPv6Address	This value is written to DMO attribute Network_Address_ 128_Bit; see Table 10
	2	Assigned_DL_Address_16 _Bit	Type: DL16Address	This value is written to DMO attribute DL_Address_16_Bit; see Table 10
	3	Assigned_Device_Role	Type: BitArray16	This value is written to DMO attribute Assigned_Device_Role; see Table 10
	4	System_Manager_ Network_Address_128_Bit	Type: IPv6Address	This value is written to DMO attribute System_Manager_ 128_Bit_Address; see Table 10
	5	System_Manager_ DL_Address_16_Bit	Type: DL16Address	This value is written to DMO attribute System_Manager_ DL_Address_16_Bit; see Table 10
	6	System_Manager_EUI64	Type: EUI64Address	This value is written to DMO attribute System_Manager_EUI64Address; see Table 10
	7	MIC	Type: OctetString4	This value is used for protecting argument 1 through 6. This MIC value is generated by the Security Manager. See 7.4.4.3.2
	8	Assigned_Max_TSDU_Size	Type: Unsigned16	Indicates the maximum TSDU supported in octets which can be converted by the source into max APDU size by taking into account the TL, security, AL headers and TMIC sizes

Table 24 – System_Manager_Contract method

Standard object type name: Device management service object (DMSO)				
Standard object type identifier: 103				
Method name	**Method ID**	**Method description**		
System_Manager_Contract	2	Method to send contract request of a new device to system manager and get contract response. Contracts are described in 6.3.11.2		
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	EUI64	Type: EUI64Address	EUI64Address of the new device trying to join the network
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Contract_Response	Type: New_Device_Contract_Response (see Table 31)	Contract response to support future communication from new device to system manager; contracts are described in 6.3.11.2
	2	MIC	Type: OctetString4	This value is used for protecting argument 1 through 6 with the join key. This MIC value is generated in Security Manager. See 7.4.4.3.2

When the DMSO generates the System_Manager_Join and System_Manager_Contract responses, it first sends these responses to the PSMO which in turn sends them to the security manager. The security manager shall protect these responses with a MIC field using the join key and send them back to the PSMO which in turn hands them back to the DMSO. The DMSO shall then send the responses back to the advertising router. This interaction with the PSMO is described in 7.4.

6.3.10 System time services

6.3.10.1 General

The time in this standard is based on international atomic time (TAI) as the time reference. The time in this standard is reported as elapsed seconds since 1958/01/01 00:00:00.

The system supports time synchronization so that, at the device level, applications may use time to coordinate activities or time-stamp information, improving energy use and reliability. System time shall be available from at least one device (system time source) on the network.

A TAI time source is not required for operation of a network compliant with this standard. Alternative time sources are converted to TAI units. The time base used by the network shall be within ±1 second of actual TAI time. The gateway shall convert nominal network TAI time to the local system time reference if it is available.

The system manager shall configure at least one system time source in each D-subnet of the network. The system manager itself may be the system time source. This is described in

6.3.10.3. The system manager shall also configure the distribution topology for the dissemination of time in the D-subnet and for the synchronization of device clocks. The system manager configures each device in the D-subnet with the clock parent(s) that the device shall use for synchronizing its clock. The DL in each device is responsible for measuring time and keeping the clocks synchronized. This is described in 9.1.9.

Backbone routers are expected to use either proprietary or standardized techniques for maintaining time synchronization. These techniques are not specified by this standard.

Devices needing to convert TAI time to hh:mm:ss format, such as on a user display, may account for a coordinated universal time (UTC) accumulated leap second adjustment.[6] The system manager shall provide this UTC adjustment to these devices. If the device needs this UTC adjustment information from the system manager, it should refresh it infrequently but periodically, such as at the start of each month or any other arbitrary clock boundary.

All devices in a network compliant with this standard share the TAI time reference with variable degrees of accuracy. To support sequence of events or other timing related operations of the application processes, all routing devices within a network compliant with this standard should be accurate to within ±10 ms. The exact clock accuracy requirement for each routing device is described in 9.1.9.2.2.

The system manager is responsible for coordinating the time across different D-subnets by selecting the appropriate system time sources in each D-subnet. This coordination is not specified by this standard.

To support sequence of events or other timing related operations of the application processes, backbone routers also should be accurate to within ±10 ms. Neither conversion of the time units used by the backbone routers nor adjustment to align with the TAI time being used by the devices compliant with this standard are specified by this standard.

If the system manager is part of the D-subnet, then the DL in the system manager is responsible for measuring time and keeping the device clock synchronized. If the system manager is connected to the backbone, it is expected to maintain clock synchronization with the rest of the network and maintain time information in TAI units. In this case, the techniques for doing so are not specified by this standard.

Protocol layers that require the current TAI time of the device may obtain it from the DMO in the DMAP.

6.3.10.2 Device clock accuracy capabilities

The system manager needs to know the clock accuracy of each device. For example, it needs to know whether a device is capable of maintaining ±1 ms accuracy for 30 s without a clock update. Such information about a device shall be provided to the system manager by the DLMO. This is described in Table 147.

6.3.10.3 System time source selection

The device implementing the system manager role may also implement the system time source role in a network, or it may delegate the system time source role to any device(s) in the network capable of playing this role as indicated by the Device_Role_Capability attribute in the DMO of the device. The system manager shall use the Assigned_Device_Role attribute in the DMO of the device to configure it as a system time source. The system manager may select the system time source based on the clock accuracy capabilities of the device.

---

6 A list of such adjustments is maintained at ftp://maia.usno.navy.mil/ser7/tai-utc.dat .

The system time source is the ultimate source of the time sense in a D-subnet. The system time source within a D-subnet shall be accurate to within ±1 s of actual TAI time, and shall monotonically increase at a rate that tracks TAI time with a maximum error of 1×10^{-6}, i.e., the rate of increase of time shall be relatively precise, even if the time source itself is relatively inaccurate.

If multiple system time sources exist within a D-subnet, they should track each other within 0,1 ms. If 0,1 ms synchronization among D-subnet system time sources cannot be arranged, the system manager shall dictate when a device switches from one system time source to the other. The dlmo.ClockStale attribute described in 9.1.9.2.3 and 9.4.2.14 shall be used to inform the device when to switch over to the other system time source. Time propagation paths are described in 6.3.10.4. If devices are to switch system time sources, the time propagation paths can be re-arranged by the system manager as appropriate.

The system manager shall ensure that each D-subnet has at least one system time source. The following are some examples of system time sources.

- In an outdoor application, the system manager may designate a few devices with global positioning system (GPS) capabilities as system time sources. Time may be propagated through the D-subnets from these sources.
- In a large network with an Ethernet backbone, the backbone itself may provide a time service that is synchronized to within 0,1 ms to a shared time reference for devices that are on the backbone. The system manager may designate the backbone routers as system time sources, and time may be propagated from these backbone routers to devices in the D-subnets.
- The system manager may periodically synchronize to a remote time source via a long-distance wired or wireless connection. This time source provides ±1 s accuracy. The system manager may then act as the system time source in the network.

If multiple system time sources exist in a D-subnet, the system manager may assign one of the system time sources as the default and the others as back-ups. The techniques for such assignment are not specified by this standard.

Clock corrections within a system time source are usually applied at a rate that can ensure that the correction does not exceed 0,5 ms in a given 30 s period. Discontinuous clock corrections are supported, with devices on a D-subnet being instructed to adjust their clocks at a specific time. This is described in 9.1.9.3.6.

6.3.10.4 Time distribution topology

In addition to system time sources, the system manager also configures clock recipients and clock repeaters in a D-subnet.

All devices in a D-subnet, except for system time sources, are configured as clock recipients, i.e., they receive periodic clock updates from one or more clock sources in their immediate neighborhoods. A clock source may be a system time source or a clock repeater.

Clock repeaters are clock recipients that also act as clock sources to certain neighbors. Clock repeaters propagate time through a D-subnet. The clock accuracy requirement for a clock repeater is described in 9.1.9.2.2.

Clock source/recipient relationships, and thus time distribution topologies, are defined by the system manager. Time propagation paths in these topologies usually match routing graphs, but this is not required. Circular time propagation paths are not allowed. Clock propagation may be arranged so that clock repeaters provide updates to their recipients soon after they themselves receive their updates.

The system manager uses the DLMO of a device to configure its clock source(s). The time of a clock recipient may be updated during each interaction with a designated clock source. The selection of clock sources and the timing of clock updates are arranged by the system manager. These clock updates are described in 9.1.9.2.

6.3.10.5 Monitoring of time synchronization accuracy

System management may support mechanisms for gathering information about the accuracy of the distributed time sense, as well as for producing an alert when the sense of time between a pair of devices varies enough to cause problems within the system. The alerts described in 9.6 may be used for this purpose by the system manager.

Vendor-specified attributes in the DMO of a device may be used for gathering such information. Vendor-specified alerts from the DMO may be used for diagnosing problems related to clock synchronization and clock maintenance.

6.3.10.6 System time service object

The system manager contains the system time service object (STSO), which shall provide the UTC accumulated leap second adjustment to the devices in the network. Other vendor-specified attributes may be added to the STSO.

The attributes of the STSO in the system manager are given in Table 25.

NOTE For more information on this leap second adjustment, see https://en.wikipedia.org/wiki/Leap_second .

Table 25 – Attributes of STSO in the system manager

Standard object type name: System time service object (STSO)				
Standard object type identifier: 100				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Current_UTC_Adjustment	1	The current value of the UTC accumulated leap second adjustment	Type: Integer16 Classification: Dynamic Accessibility: Read only Default value: 35	Devices that need to convert TAI time to hh:mm:ss format need this adjustment from the system manager; units in seconds; note that the adjustment can be negative; note that UTC and TAI are based on different start dates but this difference is not covered by this attribute; on 2012.06.30 23:59:60 the value changed from 34 s to 35 s. The mechanism used by the system manager to obtain this adjustment is not specified
Next_UTC_Adjustment_Time	2	The TAI time when the UTC adjustment value will change from the current one	Type: TAITimeRounded Classification: Dynamic Accessibility: Read only Default value: See description	If the system manager knows the next time this UTC adjustment value will change, the SM is expected to indicate this time in TAI units. If the system manager does not know this time, it is expected to indicate the current TAI time and as a result the value of the Next_UTC_Adjustment attribute shall be same as the value of the Current_UTC_Adjustment
Next_UTC_Adjustment	3	The next value of the UTC accumulated leap second adjustment	Type: Integer16 Classification: Dynamic Accessibility: Read only Default value: 35	The UTC adjustment that will go into effect at the time specified by the Next_UTC_Adjustment_Time attribute
Reserved for future editions of this standard	4..63	—	—	—
NOTE UTC and TAI and GPS-time are based on different start dates but this difference is not covered by this attribute. On 2009/01/01 the Current_UTC_Adjustment changed from 33 s to 34 s. GPS-time is always 19 s behind TAI. GPS time information includes the offset needed to convert to/from UTC.				

6.3.11 System communication configuration

6.3.11.1 General

The system manager provides control of the runtime system communication configuration. It supports configuration of the network, including attributes of the protocol suite from DL to AL.

System communication configuration includes the assignment of slots, templates, and graphs to the devices in the network. The system manager should take into account the capabilities of the device in the network while configuring such assignments. When necessary, the system should be reconfigured to recover from failure scenarios.

6.3.11.2 Contract services

6.3.11.2.1 Definition of contract

System communication configuration is achieved through the contract services provided by the system manager.

A contract refers to an agreement between the system manager and a device in the network that shall involve the allocation of network resources by the system manager to support a particular communication need of this device. This device is the source of the communication messages and the device it wants to communicate with is the destination.

A contract shall establish and support the communication path between devices in the network that are compliant with this standard to support the communication need of an application process. An application process that requires communication with an application process in another device shall request a contract. Such contract requests may originate from an application process in any one of the devices compliant with this standard, such as a field device, gateway, backbone router, or system manager.

As shown in Figure 27, contracts shall be established by the system manager. They shall also be maintained, modified and terminated by the system manager. The system manager shall interact with the affected devices in the network to perform each of these operations.

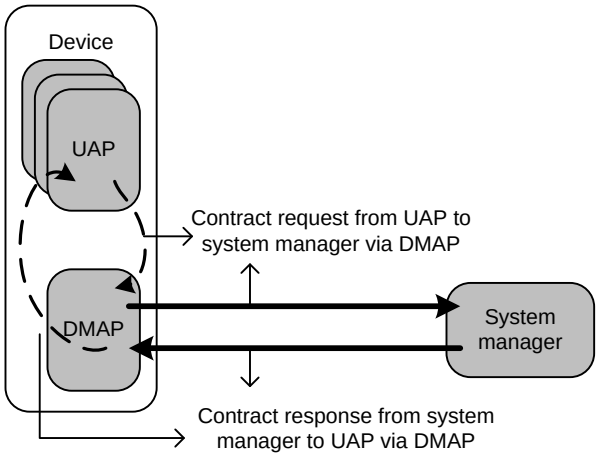


Figure 27 – UAP-system manager interaction during contract establishment

6.3.11.2.2 Directionality of contract

Contracts shall be unidirectional, i.e., a particular contract is limited to the communication from a source to a destination. For communication in the opposite direction, a separate contract shall be established.

For a two-way communication between two devices, each device shall obtain an independent contract from the system manager in order to send its messages to the other device. The peer application processes in these devices are expected to be configured such that they establish these contracts before commencing messaging in either direction. Such configurations are done either by the system manager if the application processes are the DMAPs or by host applications on plant networks or by handheld maintenance tools if the application processes are UAPs.

6.3.11.2.3 Definition of contract identifier

A contract ID (contract identifier) is a system manager-assigned identifier that shall be provided to the source after the necessary network resources have been allocated to provide the requested communication support.

The contract ID is relevant at the source, as it is used by the system manager to inform each protocol layer in the source how to treat service data units. The layers need this information before transmitting SDUs to the destination through the network. The contract requesting application process shall retrieve the assigned contract ID and shall use it to send protocol data units down the protocol suite. Protocol suite configurations at each layer for treating such

upper layer PDUs shall be referenced to the contract ID. More details are provided in 6.3.11.2.9.

Contract IDs are unique only with respect to the source, i.e., the system manager may assign the same contract ID numerical value to two independent devices to support their independent contract requests. The combination of a source IPv6Address and its contract ID shall be unique across the network.

The contract ID is also relevant at the backbone router that supports communication intended for a destination in the D-subnet supported by that backbone router. This is because the DL in the backbone router needs to determine how to send the NPDU through the D-subnet to its destination. Configuration of the DL in the backbone router for treating such NPDUs shall be referenced to the combination of the source IPv6Address and its contract ID. More details are provided in 6.3.11.2.9.2.

The contract ID is not relevant at any other intermediate device along the path between the source and the destination.

While contract IDs are 2-octet values, the system manager shall restrict the assignment of contract IDs that fall within the range of 1..255 to contracts involving one or more field devices, since such devices usually have tighter memory constraints than other devices, thus enabling such field devices to store contract IDs using only one octet. Contract ID 0 is reserved to mean noContract.

6.3.11.2.4 Architecture supporting contract-related messaging

6.3.11.2.4.1 General

The DMO in each device and the SCO in the system manager shall work together to provide contract related services such as contract establishment, contract maintenance and modification, and contract termination to each device.

6.3.11.2.4.2 Handling contract-related services within device

The DMO in each device shall be responsible for requesting, maintaining, modifying, and terminating each and every contract assigned to that device.

Any application process that requires a contract needs to send the request to the DMO which in turn shall send the request to the system manager. After the contract has been established, this application process shall not try to use network resources in excess of the ones allocated for this contract. After the contract has been established, this application process may later request for a modification or termination of this contract as appropriate.

The Contract_Table structured attribute of the DMO may be accessed directly or indirectly by the SCO present in the system manager.

The system manager indirectly accesses the Contract_Table structure attribute when it sends any contract related response to the device. The device shall update the Contract_Table structured attribute of the DMO every time a contract response is received from the system manager. A new entry in the Contracts_Table structured attribute of the DMO shall be created every time a contract response associated with the successful creation of a new contract is received from the system manager.

The system manager may directly read or write any element present in the Contract_Table structured attribute of the DMO once the contract entry exists in the device.

6.3.11.2.4.3 Handling contract related services in the network

The SCO in the system manager is responsible for establishing, maintaining, modifying, and terminating all contracts in the network. The SCO shall coordinate with the DMO of each device to perform these operations.

6.3.11.2.4.4 System communication configuration object

The attributes of the SCO are given in Table 26. The methods of the SCO are given in Table 27.

Table 26 – Attributes of SCO in the system manager

Standard object type name: System communication configuration object (SCO)				
Standard object type identifier: 102				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Reserved for future editions of this standard	1..63	—	—	—

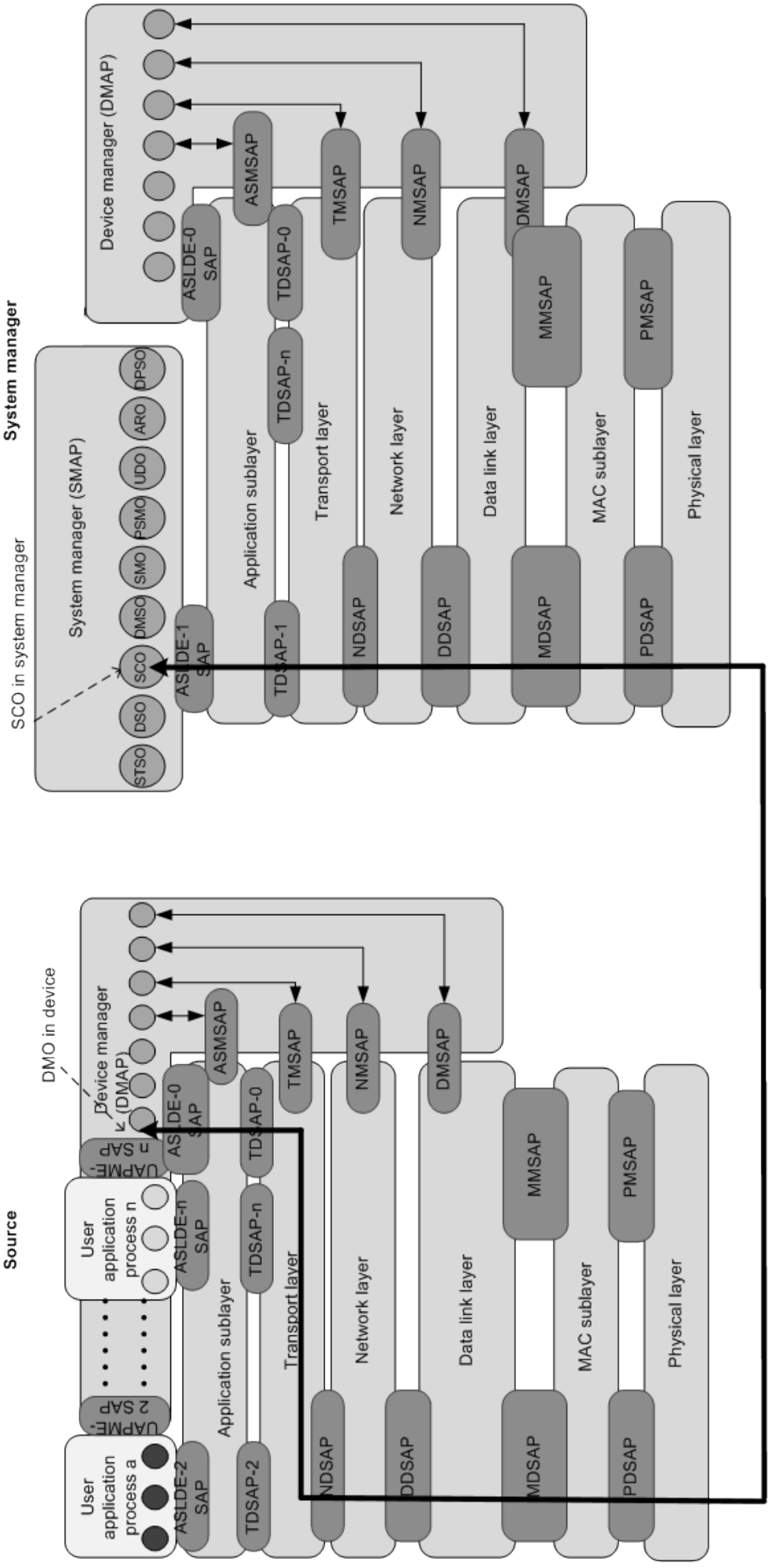


Figure 28 – Contract-related interaction between DMO and SCO

6.3.11.2.4.5 Contract-related messages

All contract-related messages between the SCO in the system manager and the DMO of the device shall be application level client/server messages (i.e., writes and reads on standard object attributes and executes on standard object methods). This is illustrated in Figure 28.

Contract-related messages include contract requests and contract responses; these are described in 6.3.11.2.5.4.

6.3.11.2.5 Contract establishment

6.3.11.2.5.1 General

Contracts shall be established by the system manager when it receives a contract request. An application process, which needs to communicate with a peer process across the network, issues a contract request to the DMAP within the device. The DMO in this requesting device shall send this contract request to the SCO in the system manager. Each contract request shall include arguments that are used by the SCO to determine the network resource allocation necessary to support this request. These arguments are discussed in 6.3.11.2.5.4.

If a device receives a service request but does not already have a contract needed in order to send the service response, it should request a contract. The device shall not send the service response until the contract response is received from the SCO of the system manager and all resources needed to support the contract are successfully configured.

The algorithms used by the SCO to determine the necessary allocation of network resources are not specified in this standard, as they are all internal to the system manager. Vendors are expected to implement algorithms in the system manager that can determine the necessary allocation of network resources for the contract requests sent by the devices being managed by that system manager.

Based on this determination, the SCO shall allocate the network resources by communicating with the necessary devices in the network and providing necessary protocol suite configurations to each one of them. This shall include the configuration of the destination and the source. Details are provided in 6.3.11.2.6. Contracts shall be established for both scheduled and unscheduled communication between applications.

As part of the configuration of the source, the SCO shall also provide the contract ID. When the source receives this configuration, it shall use this contract information to start transmitting the TSDUs that needed this communication support. After the contract has been established, the source shall not try to use network resources in excess of the ones allocated for this contract.

6.3.11.2.5.2 Relation between contracts and sessions

Sessions shall be established between the T-port in the source and the corresponding T-port in the destination. All communication between these ports shall be secured using a T-key that is issued by the security manager. Session establishment is described in 7.5. Contracts shall support the communication between peer application processes that reside on top of these T-ports in the protocol stack.

Multiple contracts may be established between these peer application processes to support different communication needs. As each application process in a device is associated with a T-port, all these contracts of these peer application processes shall use the same T-ports in the source and destination and so the same T-key shall be used for securing all the communication that occurs using these multiple contracts.

The T-key between the corresponding T-ports in the source and the destination needs to be established before a contract can be used to send messages through these T-ports. So,

before the DMO sends the contract request to the SCO it is expected to check with the DSMO in the DMAP to see if a T-key exists between the corresponding T-ports in the source and the destination. If such a T-key does not exist, the DSMO is expected to send a T-key request to the security manager and obtain a new T-key. This T-key request is described in 7.6.3. If a T-key already exists between the corresponding T-ports, the DMO can send the contract request to the SCO immediately.

If an existing T-key of a particular T-port in the device is terminated for some reason by the security manager or by the device itself, any contract that uses this particular T-port will fail as the message will not get sent down the protocol stack in the device. In this case, TL is expected to send back a failure indication to the application process that generated the message. This application process in turn is expected to send a contract request to the DMO which checks with the DSMO for a T-key. The DSMO may send a new T-key request to the security manager if necessary.

6.3.11.2.5.3 Devices that can request contracts

Only a traffic source shall submit a contract request to the system manager. Other devices in the network are not allowed to submit a contract request on behalf of the source.

6.3.11.2.5.4 Contract request and response arguments

Contract request arguments are pieces of information that are provided by the contract requesting application process. They are based on the communication need that the requesting application process is interested in.

Applications should be designed such that they request only the least amount of communication support needed to satisfy their communication needs. For example, an application that needs to publish periodic messages every 1 minute should not request a 10 second period.

The contract response arguments shall include the contract ID and other information. This information is intended to provide the basic protocol suite configuration necessary at the source.

The arguments included in the contract request and response messages are described in Table 27.

Table 27 – SCO method for contract establishment, modification, or renewal *(1 of 8)*

Standard object type name: System communication configuration object (SCO)				
Standard object type identifier: 102				
Method name	**Method ID**	**Method description**		
Contract_ Establishment _ Modification_ Renewal	1	Method to establish a new contract / modify an existing contract / renew an existing contract by sending request to system manager		
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Contract_Request_ID	Unsigned8	A numerical value, uniquely assigned by the device sending the request to the system manager, to identify the request being made. Defaults to zero, and resets to zero. Increments with each use. This ID shall be repeated if exactly the same request is re-sent due to lack of response. Rolls over to zero
	2	Request_Type	Unsigned8	Type of contract request sent to the system manager. Named values: 0: new contract 1: contract modification 2: contract renewal Some of the input arguments below are not applicable based on this argument value; see Table 28 for details
	3	Contract_ID	Unsigned16	Existing contract ID that needs to be modified or renewed
	4	Communication_ Service_Type	Unsigned8	Type of communication service for which the contract is being requested. Named values: 0: periodic / scheduled 1: aperiodic / unscheduled Some of the input arguments below are not applicable based on this argument value; see Table 28 for details
	5	Source_SAP	Unsigned16	TDSAP in the source that will be using this contract, once it is assigned, to send application messages down the protocol stack
	6	Destination_Address	IPv6Address	The address of the device that the source wants to send application messages to; note that this information may be provided to the source during provisioning or during configuration of the application process

Table 27 *(2 of 8)*

Standard object type name: System communication configuration object (SCO)				
Standard object type identifier: 102				
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	7	Destination_SAP	Unsigned16	TDSAP in the destination that will be used to send these messages to the AL; note that this information may be provided to the source during provisioning or during configuration of the application process
	8	Contract_Negotiability	BitString8	Determines if the system manager can change the requested contract to meet the network resources available and if the system manager can revoke this contract to make resources available to higher priority contracts. Named indices: 0: not revocable; 1: non-negotiable; 2..7: reserved Contract negotiability is described in 6.3.11.2.7.2
	9	Contract_Expiration_Time	Unsigned32	Determines how long the system manager should keep the contract before it is terminated; units in seconds
	10	Contract_Priority	Unsigned8	Requests a base priority for all messages sent using the contract. Named values: 0: best effort queued; 1: real time sequential; 2: real time buffer; 3: network control Contract priority is described in 6.3.11.2.7.3
	11	Payload_Size	Unsigned16	Indicates the maximum payload size in octets (represented as APDU size) that the source is interested in transmitting. Value range: 3..1 252

Table 27 *(3 of 8)*

Standard object type name: System communication configuration object (SCO)				
Standard object type identifier: 102				
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	12	Reliability_And_Publish DoNotAutoRetransmit	Unsigned8	PublishDoNotAutoRetransmit: Bit 0 indicates whether retransmission of old publish data is not supported because the prior buffer value is always overwritten with new publish data. Bit 0 is only applicable for periodic communication and has value 0 for aperiodic communication (see 12.12.2 for description). [a] Reliability: Bits 1..7 indicate the supported reliability for delivering the transmitted APDUs to the destination. Bit 0: Unsigned1: Named values: 0: auto-retransmit; 1: do not auto-retransmit. Bits 1..7: Unsigned7: Named values: 0: low; 1: medium; 2: high
	13	Requested_Period	Integer16	Used for periodic communication to identify the desired publishing period in the contract request. Valid range: A value of $N > 0$ specifies a period of N s, while $N < 0$ specifies a period of -1/N s. $N = 0$ disables the communication
	14	Requested_Phase	Unsigned8	Used for periodic communication to identify the desired phase (within the publishing period) of publications in the contract request. Valid range: 0..99; any other value indicates that device only cares about period and does not care about phase. See 6.3.11.2.7.4
	15	Requested_Deadline	Unsigned16	Used for periodic communication to identify the maximum end-to-end transport delay desired. Unit:10 ms

Table 27 *(4 of 8)*

Standard object type name: System communication configuration object (SCO)				
Standard object type identifier: 102				
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	16	Committed_Burst	Integer16	Used for aperiodic communication to identify the long term rate that needs to be supported for client/server or source/sink messages. Valid range: A value of $N > 0$ specifies a mean rate of APDUs per second, while $N < 0$ specifies a mean rate of -1/N APDUs per second. $N = 0$ is invalid.
	17	Excess_Burst	Integer16	Used for aperiodic communication to identify the short term rate that needs to be supported for client/server or source/sink messages. Valid range: see input argument 16
	18	Max_Send_Window_ Size	Unsigned8	Used for aperiodic communication to identify the maximum number of client requests that may be simultaneously awaiting a response
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Contract_Request_ID	Unsigned8	The input argument Contract_Request_ID that was received in the corresponding contract request is used as this output argument

Table 27 *(5 of 8)*

Standard object type name: System communication configuration object (SCO)				
Standard object type identifier: 102				
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	2	Response_Code	Unsigned8	Indicates if the system manager was successful or not in supporting the contract request; indicates if the source can use the contract immediately or if it has to wait; also indicates if the requested communication is being supported as is or if the system manager negotiated the request down. Named values: 0: success with immediate effect; 1: success with delayed effect; 2: success with immediate effect but negotiated down; 3: success with delayed effect but negotiated down; 4: failure with no further guidance; 5: failure with retry guidance; 6: failure with retry and negotiation guidance. Depending on the value of this argument, some of the output arguments below are not applicable. See Table 29 for details and 6.3.11.2.12 for failure scenarios
	3	Contract_ID	Unsigned16	A numeric value uniquely assigned by the system manager to the contract being established and sent to the source. Contract IDs are unique per device. Depending on the requested resources, multiple contract request IDs from a device may be mapped to a single contract ID. In the device, the contract ID is passed in the DSAP control field of each layer and is used to look up the contracted actions that shall be taken on the associated PDU as it goes down the protocol suite at each layer (value 0 reserved to mean no contract)
	4	Communication_Service_Type	Unsigned8	Type of communication service supported by this contract. Unsigned8: see input argument 4. Some of the output arguments below are not applicable based on this argument value; see Table 29 for details
	5	Contract_Activation_Time	TAINetwork Time	Start time for the source to start using the assigned contract

Table 27 *(6 of 8)*

Standard object type name: System communication configuration object (SCO)				
Standard object type identifier: 102				
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	6	Assigned_Contract_ Expiration_Time	Unsigned32	Determines how long the system manager shall keep the contract before it is terminated. Units: s
	7	Assigned_Contract_ Priority	Unsigned8	Establishes a base priority for all messages sent using the contract. See input argument 10. Contract priority is described in 6.3.11.2.7.3
	8	Assigned_Max_TSDU_ Size	Unsigned16	Indicates the maximum TSDU in octets which can be converted by the source into max APDU size supported by taking into account the TL, security, AL header and TMIC sizes. Valid range: 70..1 280. The system manager shall take into account the Max_NSDU_Size constant attribute reported by the NLMOs of the source and the destination (see Table 206) while determining the value of this argument. Fragmentation is done at the NL if the NPDU exceeds the max size of a DSDU. Fragmentation and reassembly are described in 10.2.5
	9	Assigned_Reliability_ And_PublishDoNot AutoRetransmit	Unsigned8	See input argument 12
	10	Assigned_Period	Integer16	See input argument 13
	11	Assigned_Phase	Unsigned8 Valid range: 0..99	Used for periodic communication to identify the assigned phase (within the publishing period) of publications in the contract
	12	Assigned_Deadline	Unsigned16	Used for periodic communication to identify the maximum end-to-end transport delay supported by the assigned contract. Unit: 10 ms
	13	Assigned_Committed_ Burst	Integer16	Used for aperiodic communication to identify the long term rate that is supported for client/server or source/sink messages. Valid range: A value of $N > 0$ specifies a mean rate of APDUs per second, while $N < 0$ specifies a mean rate of $-1/N$ APDUs per second. $N = 0$ is invalid

Table 27 *(7 of 8)*

Standard object type name: System communication configuration object (SCO)				
Standard object type identifier: 102				
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	14	Assigned_Excess_Burst	Integer16	Used for aperiodic communication to identify the short term rate that is supported for client/server or source/sink messages. Valid range: A value of $N > 0$ specifies a mean rate of APDUs per second, while $N < 0$ specifies a mean rate of -1/N APDUs per second. $N = 0$ is invalid
	15	Assigned_Max_Send_ Window_Size	Unsigned8	Used for aperiodic communication to identify the allowed maximum number of client requests that can simultaneously await a response
	16	Retry_Backoff_Time	Unsigned16	Used in the case of response code = failure with retry guidance or failure with retry and negotiation guidance; indicates the amount of time the source should back off before resending the contract request; units in seconds; failure scenarios are described in 6.3.11.2.12
	17	Negotiation_Guidance	BitString8	Used in the case of response code = failure with retry and negotiation guidance; indicates the Contract_Negotiability value supportable by system manager. Index assignments: see input argument 8 Failure scenarios are described in 6.3.11.2.12
	18	Supportable_Contract_ Priority	Unsigned8	Indicates the base priority supportable by the system manager for all messages sent using the contract. Unsigned8: see input argument 10
	19	Supportable_max_ TSDU_Size	Unsigned16 Valid range: 70..1 280	Indicates the maximum NSDU supportable by the system manager; units in octets.
	20	Supportable_Reliability_ And_PublishDoNot AutoRetransmit	Unsigned8	See input argument 12
	21	Supportable_Period	Integer16	Used for periodic communication to identify the supportable publishing period by the system manager. Valid range: see input argument 13
	22	Supportable_Phase	Unsigned8 Valid range: 0..99	Used for periodic communication to identify the phase (within the publishing period) of publications supportable by the system manager

Table 27 *(8 of 8)*

Standard object type name: System communication configuration object (SCO)				
Standard object type identifier: 102				
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	23	Supportable_Deadline	Unsigned16	Used for periodic communication to identify the maximum end-to-end transport delay supportable by the system manager. Unit:10 ms
	24	Supportable_Committed_Burst	Integer16	Used for aperiodic communication to identify the long term rate that can be supported for client/server or source/sink messages. Valid range: see input argument 16
	25	Supportable_Excess_Burst	Integer16	Used for aperiodic communication to identify the short term rate that can be supported for client/server or source/sink messages. Valid range: A value of $N > 0$ specifies a mean rate of APDUs per second, while $N < 0$ specifies a mean rate of -1/N APDUs per second. $N = 0$ is invalid
	26	Supportable_Max_Send_Window_Size	Unsigned8	Used for aperiodic communication to identify the supportable maximum number of client requests that can simultaneously await a response
[a] The coding of this attribute is the inverse of the related attribute 7 of Table 265.				

Table 27 also contains input and output arguments that shall be used for contract modification and contract renewal. Contract modification and contract renewal are discussed in 6.3.11.2.11.

Table 27 also contains output arguments that shall be used for failure scenarios when the system manager is not able to support the contract request. These failure scenarios are discussed in 6.3.11.2.12.

Some of the input arguments in Table 27 are not applicable when the Request_Type and/or Communication_Service_Type arguments are given certain values and so shall not be included in the request. This information is provided in Table 28.

Table 28 – Input argument usage for SCO method for contract establishment, modification, or renewal

Input argument	Not applicable for	
	Request_Type value	Communication_Service_Type value
Contract_Request_ID	—	—
Request_Type	—	—
Contract_ID	0	—
Communication_Service_Type	—	—
Source_SAP	—	—
Destination_Address	—	—
Destination_SAP	—	—
Contract_Negotiability	—	—
Contract_Expiration_Time	—	—
Contract_Priority	—	—
Payload_Size	—	—
Reliability_And_PublishDoNotAutoRetransmit	—	—
Requested_Period	—	1
Requested_Phase	—	1
Requested_Deadline	—	1
Committed_Burst	—	0
Excess_Burst	—	0
Max_Send_Window_Size	—	0

Some of the output arguments in Table 27 are not applicable when the Response_Code and/or Communication_Service_Type arguments are given certain values and so shall not be included in the response. This information is provided in Table 29.

Table 29 – Output argument usage for SCO method for contract establishment, modification, or renewal

Output argument	Not applicable for	
	Response_Code value	Communication_Service_Type value
Contract_Request_ID	—	—
Response_Code	—	—
Contract_ID	4, 5, 6	—
Communication_Service_Type	4, 5	—
Contract_Activation_Time	0, 2, 4, 5, 6	—
Assigned_Contract_Expiration_Time	4, 5, 6	—
Assigned_Contract_Priority	4, 5, 6	—
Assigned_Max_TSDU_Size	4, 5, 6	—
Assigned_Reliability_And_PublishDoNotAutoRetransmit	4, 5, 6	—
Assigned_Period	4, 5, 6	1
Assigned_Phase	4, 5, 6	1
Assigned_Deadline	4, 5, 6	1
Assigned_Committed_Burst	4, 5, 6	0
Assigned_Excess_Burst	4, 5, 6	0
Assigned_Max_Send_Window_Size	4, 5, 6	0
Retry_Backoff_Time	0, 1, 2, 3, 4	—
Negotiation_Guidance	0, 1, 2, 3, 4, 5	—
Supportable_Contract_Priority	0, 1, 2, 3, 4, 5	—
Supportable_max_TSDU_Size	0, 1, 2, 3, 4, 5	—
Supportable_Reliability_And_PublishDoNotAutoRetransmit	0, 1, 2, 3, 4, 5	—
Supportable_Period	0, 1, 2, 3, 4, 5	1
Supportable_Phase	0, 1, 2, 3, 4, 5	1
Supportable_Deadline	0, 1, 2, 3, 4, 5	1
Supportable_Committed_Burst	0, 1, 2, 3, 4, 5	0
Supportable_Excess_Burst	0, 1, 2, 3, 4, 5	0
Supportable_Max_Send_Window_Size	0, 1, 2, 3, 4, 5	0

6.3.11.2.6 Protocol suite configuration

6.3.11.2.6.1 General

As part of contract establishment, the SCO shall configure the necessary devices in the network by providing necessary protocol suite configurations to each one of them. This shall include the configuration of the destination and the source, as illustrated in Figure 29.

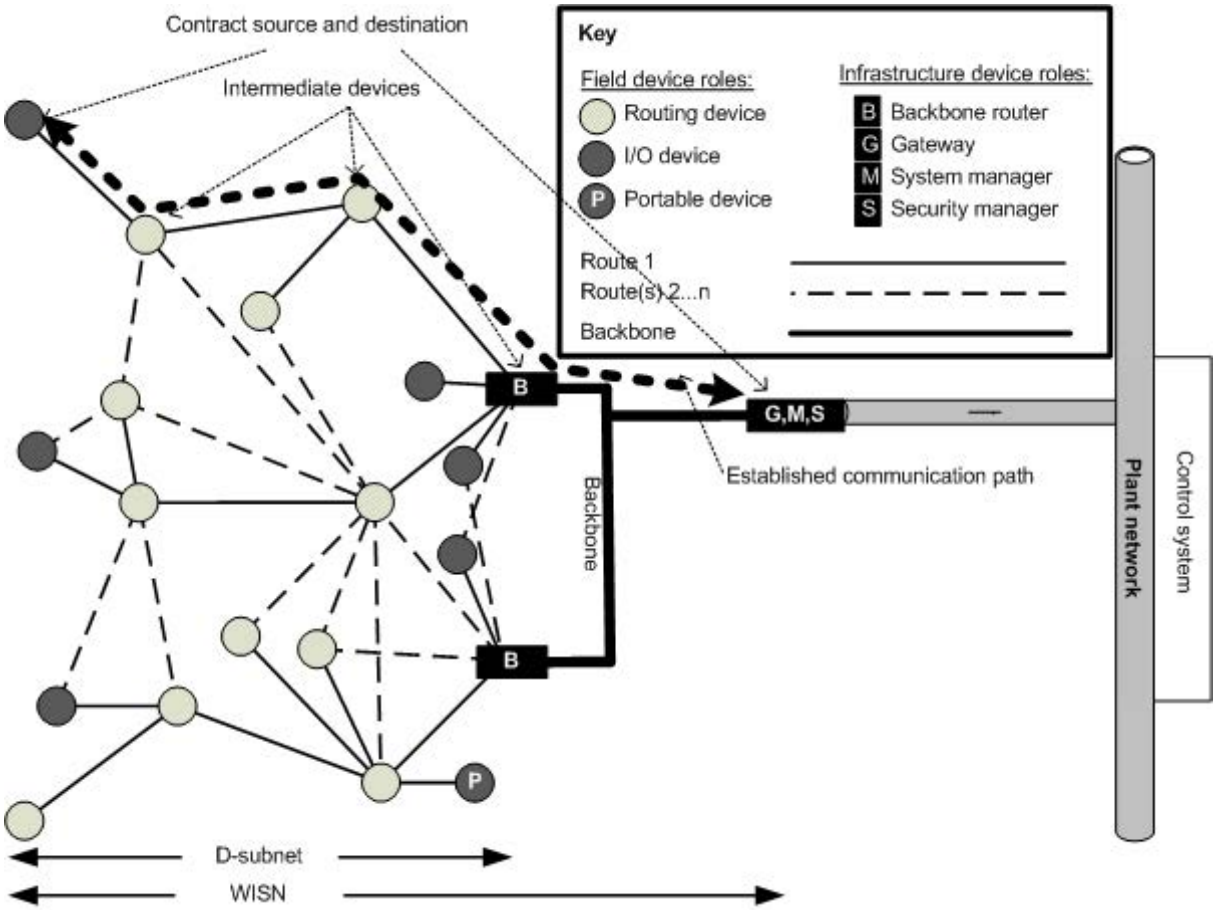


Figure 29 – Contract source, destination, and intermediate devices

Intermediate devices in the network that support the communication path being established between the source and the destination shall be configured by the SCO. Such intermediate devices along the path may include both field routers and backbone routers.

6.3.11.2.6.2 Configuration of intermediate field routers

Configuration of intermediate field routers shall be limited to the DLE in each field router, as the message from the source to the destination traverses only through the DLE of each field router along the path.

Attributes and methods defined for the DLMO of the field routers shall be used by the system manager to configure the intermediate field routers.

6.3.11.2.6.3 Configuration of intermediate backbone routers

Configuration of intermediate backbone routers shall be limited to the NL and, in some cases, the DLE in each backbone router, as the message from the source to the destination traverses through the DLE in the case of backbone routers that belong to the corresponding source and destination D-subnets, and the NLE of each backbone routers along the path.

Attributes and methods defined for the DLMO and NLMO of the backbone routers shall be used by the system manager to configure the intermediate backbone routers.

6.3.11.2.6.4 Configuration of destination

Configuration of destination shall include the configuration of all the protocol layers. The attributes and methods defined for the DLMO, NLMO, and TLMO shall be used by the system manager to configure the destination.

6.3.11.2.6.5 Configuration of source

The output arguments described in Table 27 are used at various layers of the source to determine the treatment of PDUs belonging to this contract.

The attributes and methods defined for the DLMO, NLMO, and TLMO shall be used by the system manager to configure the source.

A contract response shall be sent to the source either after all necessary network resources have been configured or after the system manager determines the time it would take to configure all necessary network resources. Depending on the situation, the system manager shall indicate if the assigned contract can be used with immediate effect or with delayed effect. The message sequence diagram in Figure 30 illustrates the case of immediate effect.

After the contract has been established, the source shall not try to use network resources in excess of the ones allocated for this contract.

6.3.11.2.6.6 Contract information in device management object

The DMO in the source shall maintain a list of all assigned contracts using the Contracts_Table attribute. This attribute shall be based on the data structure Contract_Data. When a new contract gets established, a new row shall be added to this Contracts_Table attribute with the relevant contract information. When an existing contract gets modified or terminated, the corresponding row shall be modified or deleted in this Contracts_Table attribute.

The SCO can also modify the parameters of the Contract_Table attributes by accessing them directly without exchanging entire Contract_Data structures.

The elements of the data structure Contract_Data are defined in Table 30.

Table 30 – Contract_Data data structure *(1 of 3)*

Standard data type name: Contract_Data		
Standard data type code: 401		
Element name	**Element identifier**	**Element type**
Contract_ID*	1	Type: Unsigned16 Classification: Static Accessibility: Read/write Named values: 0: no contract This element is the same as output argument Contract_ID in Table 27. * This element is used as the index field for methods described in Table 33 and Table 34
Contract_Status	2	Type: Unsigned8 Classification: Static Accessibility: Read only Named values: 0: success with immediate effect; 1: success with delayed effect; 2: success with immediate effect but negotiated down; 3: success with delayed effect but negotiated down This element is related to the output argument Response_Code in Table 27
Communication_Service_Type	3	Type: Unsigned8 Classification: Static Accessibility: Read/write Named values: 0: periodic / scheduled communication; 1: aperiodic / unscheduled communication This element is the same as output argument Communication_Service_Type in Table 27
Contract_Activation_Time	4	Type: TAINetworkTime Classification: Static Accessibility: Read/write This element is the same as output argument Contract_Activation_Time in Table 27
Source_SAP	5	Type: Unsigned16 Classification: Static Accessibility: Read/write This element is the same as input argument Source_SAP in Table 27
Destination_Address	6	Type: IPv6Address Classification: Static Accessibility: Read/write This element is the same as input argument Destination_Address in Table 27

Table 30 *(2 of 3)*

Standard data type name: Contract_Data		
Standard data type code: 401		
Element name	**Element identifier**	**Element type**
Destination_SAP	7	Type: Unsigned16 Classification: Static Accessibility: Read/write This element is the same as input argument Destination_SAP in Table 27
Assigned_Contract_Expiration_Time	8	Type: Unsigned32 Classification: Static Accessibility: Read/write Unit: 1 s This element is the same as output argument Assigned_Contract_Expiration_Time in Table 27
Assigned_Contract_Priority	9	Type: Unsigned8 Classification: Static Accessibility: Read/write Named values: 0: best effort queued; 1: real time sequential; 2: real time buffer; 3: network control This element is the same as output argument Assigned_Contract_Priority in Table 27
Assigned_Max_TSDU_Size	10	Type: Unsigned16 Classification: Static Accessibility: Read/write Valid range: 70..1 280 This element is the same as output argument Assigned_Max_TSDU_Size in Table 27
Assigned_Reliability_And_ PublishDoNotAutoRetransmit	11	Type: Unsigned8 Classification: Static Accessibility: Read/write Valid range: Bit 0: 0 -- i.e., always auto-retransmit [a] Bits 1..7:: Named values: 0: low; 1: medium; 2: high This element is the same as output argument Assigned_Reliability_And_PublishDoNotAutoRetransmit in Table 27

Table 30 *(3 of 3)*

Standard data type name: Contract_Data		
Standard data type code: 401		
Element name	**Element identifier**	**Element type**
Assigned_Period	12	Type: Integer16 Classification: Static Accessibility: Read/write Valid range: A value of N > 0 specifies a period of N s, while N < 0 specifies a period of -1/N s. N = 0 is invalid. This element is the same as output argument Assigned_Period in Table 27
Assigned_Phase	13	Type: Unsigned8 Classification: Static Accessibility: Read/write Valid range: 0..99 This element is the same as output argument Assigned_Phase in Table 27
Assigned_Deadline	14	Type: Unsigned16 Classification: Static Accessibility: Read/write Unit:10 ms This element is the same as output argument Assigned_Deadline in Table 27
Assigned_Committed_Burst	15	Type: Integer16 Classification: Static Accessibility: Read/write Valid range: A value of N > 0 specifies a mean rate of APDUs per second, while N < 0 specifies a mean rate of -1/N APDUs per second. N = 0 is invalid. This element is the same as output argument Assigned_Committed_Burst in Table 27
Assigned_Excess_Burst	16	Type: Integer16 Classification: Static Accessibility: Read/write Valid range: A value of N > 0 specifies a mean rate of APDUs per second, while N < 0 specifies a mean rate of -1/N APDUs per second. N = 0 is invalid. This element is the same as output argument Assigned_Excess_Burst in Table 27
Assigned_Max_Send_Window_Size	17	Type: Unsigned8 Classification: Static Accessibility: Read/write This element is the same as output argument Assigned_Max_Send_Window_Size in Table 27
[a] The coding of this attribute is the inverse of the related attribute 7 of Table 265.		

6.3.11.2.6.7 Configuration of new device

The process for a new device to join is described in 7.4. As part of the joining process for a new device, a contract between the new device and the system manager shall be established.

The new device shall use the Proxy_System_Manager_Contract method defined for the DMO of the advertising router to send this contract request, which is then forwarded to the system manager, and to get the contract response from the system manager via the advertising router. The Proxy_System_Manager_Contract method is defined in Table 20. The advertising router shall use the System_Manager_Contract method defined in the DMSO for forwarding this contract request and for receiving the contract response associated with the joining process of this new device. The System_Manager_Contract method is defined in Table 24. The DMSO works with the SCO to generate this contract response. When the new device gets this contract response, a new row shall be added to the Contracts_Table attribute in the DMO of the new device with the relevant contract information.

The output arguments in both these methods shall be based on the data structure New_Device_Contract_Response. The elements of the data structure New_Device_Contract_Response are defined in Table 31.

Table 31 – New_Device_Contract_Response data structure *(1 of 2)*

Standard data type name: New_Device_Contract_Response		
Standard data type code: 405		
Element name	**Element identifier**	**Element type**
Contract_ID	1	Type: Unsigned16 Classification: Static Accessibility: Read/write Named values: 0: no contract This element is related to the output argument Contract_ID in Table 27
Assigned_Max_TSDU_Size	2	Type: Unsigned16 Classification: Static Accessibility: Read/write Valid range: 70..1 280 This element is related to the output argument Assigned_Max_TSDU_Size in Table 27
Assigned_Committed_Burst	3	Type: Integer16 Classification: Static Accessibility: Read/write Valid range: A value of $N > 0$ specifies a mean rate of APDUs per second, while $N < 0$ specifies a mean rate of -1/N APDUs per second. $N = 0$ is invalid. This element is related to the output argument Assigned_Committed_Burst in Table 27
Assigned_Excess_Burst	4	Type: Integer16 Classification: Static Accessibility: Read/write Valid range: A value of $N > 0$ specifies a mean rate of APDUs per second, while $N < 0$ specifies a mean rate of -1/N APDUs per second. $N = 0$ is invalid. This element is related to the output argument Assigned_Excess_Burst in Table 27
Assigned_Max_Send_Window_Size	5	Type: Unsigned8 Classification: Static Accessibility: Read/write This is related to the output argument Assigned_Max_Send_Window_Size in Table 27

Table 31 *(2 of 2)*

Standard data type name: New_Device_Contract_Response		
Standard data type code: 405		
Element name	**Element identifier**	**Element type**
NL_Header_Include_Contract_Flag	6	Type: Boolean1 Classification: Static Accessibility: Read/write This is related to the corresponding element in Table 208 and is used for configuring the NL of the new device to use the contract assigned to the new device
NL_Next_Hop	7	Type: IPv6Address Classification: Static Accessibility: Read/write This is related to the corresponding element in Table 208 and is used for configuring the NL of the new device to use the contract assigned to the new device
NL_NWK_HopLimit	8	Type: Unsigned8 Classification: Static Accessibility: Read/write This is related to the corresponding element in Table 208 and is used for configuring the NL of the new device to use the contract assigned to the new device
NL_Outgoing_Interface	9	Type: Unsigned8 Classification: Static Accessibility: Read/write Named values: 0: DL; 1: backbone This is related to the corresponding element in Table 208 and is used for configuring the NL of the new device to use the contract assigned to the new device

The new device shall use the DL information provided in the advertisement DPDU to support this contract. After the device joins the network, the system manager shall access the relevant DLMO attributes in the device to modify this DL information as appropriate. More information about this DL information and the DLMO attributes is given in 9.1.14.

If the new device is not allowed by the security manager to join the network, the security manager should inform the system manager to free up this contract and the associated network resources. When so notified, the system manager shall free up the contract and the associated network resources of such a device that is not allowed to join the network.

6.3.11.2.7 Quality of service

6.3.11.2.7.1 General

The contract assigned by the system manager to a requesting application process also indicates the quality of service (QoS) for the provided communication service. The contract establishment shall be used to reach this QoS agreement between the requesting application process and the system manager.

An application process that wants to communicate with its peer may indicate the QoS desired for this communication in its contract request. The input arguments described in Table 27 may be used for this purpose.

The input arguments Contract_Priority and Payload_Size may be used in contract requests pertaining to both periodic and aperiodic communication services. Input arguments Requested_Period, Requested_Phase and Requested_Deadline are relevant for periodic communications. Input arguments Committed_Burst, Excess_Burst and Max_Send_Window_Size are relevant for aperiodic communications. The input argument Reliability_And_PublishDoNotAutoRetransmit contains information about desired reliability which is relevant for both periodic and aperiodic communications. It also indicates if the application process wants to retransmit old periodic communication data if new data is not available.

In the contract response, the system manager shall indicate the QoS level provided for the assigned communication service. The output arguments described in Table 27 corresponding to the above mentioned input arguments shall be used for this purpose.

6.3.11.2.7.2 Contract negotiability

A source that is sending a contract request shall also indicate whether the requested communication service and the QoS are negotiable, i.e., whether the system manager can assign a contract that provides a different communication service and QoS than the ones requested if it cannot support the request as is, and whether the system manager can revoke the contract if necessary. The input argument Contract_Negotiability shall be used for this purpose.

Table 27 contains arguments that are necessary for contract negotiation between the source and the system manager. If the system manager is unable to support a contract request, it may choose to provide contract negotiation guidance. Such guidance shall be provided using the output arguments in Table 27 that start with the word supportable, e.g., Supportable_Contract_Priority.

If the system manager is unable to support the contract request at the time it was received but expects to be able to support such a request in the future, it may indicate this by using the output argument Retry_Backoff_Time.

6.3.11.2.7.3 Contract priorities and message priorities

Two priority levels shall be supported in the system, contract priority and message priority.

Contract priority shall establish a base priority for all messages sent using that contract. Four contract priorities shall be supported using 2 bits as follows:

- 0b11, Network control, which may be used for critical management of the network by the system manager;
- 0b10, Real time buffer, which may be used for periodic communications in which the message buffer is overwritten whenever a newer message is generated;
- 0b01, Real time sequential, which may be used for applications such as voice or video that need sequential delivery of messages; and
- 0b00, Best effort queued, which may be used for client/server communications.

Message priority shall establish priority within a contract. Two message priorities shall be supported using 1 bit, low = 0 and high = 1. Another 1 bit is reserved for future releases of this standard and shall be set to 0.

Contract priority shall be specified by the application, during contract establishment time, in its contract request. It may be used by the system manager to establish preferred routes for

high priority contracts and for load balancing the network. The system manager shall convey the assigned contract priority to the source in the contract response.

Message priority shall be supplied by the application for every message sent down the protocol suite. In the source, the message priority shall flow down the protocol suite. The contract priority shall be added at the NL. Contract priority shall have precedence over message priority.

Combined contract/message priority shall be used to resolve contention for scarce resources when these messages are forwarded through the network. The DL shall use this information to drive queuing decisions when forwarding messages on the D-subnet. It shall be included only in the DL header. When a message is sent on a backbone, priority shall be included in the network headers. The NL shall use priority to drive queuing decisions on a backbone.

6.3.11.2.7.4 Arguments related to phase

The input argument Requested_Phase shall be used by the application process requesting the contract to request a phase which is the time offset from the beginning of a period. This time offset is expressed as a percentage of the time within a period. All periods shall be calculated such that their start times are synchronous with the beginning of TAI time. Applications may use the Requested_Phase to achieve time-synchronized, distributed loop execution with minimum latency and bounded jitter. The exact timing of the phase as it relates to the DL is specified by the link number, which is described in 9.4.3.7.

6.3.11.2.8 Contract establishment message sequence diagram

Figure 30 shows an example of a message sequence chart for the establishment of a contract. This example does not involve any timeouts and the source device accepts the contract established by the system manager even if this contract provides a different communication service than the one requested.

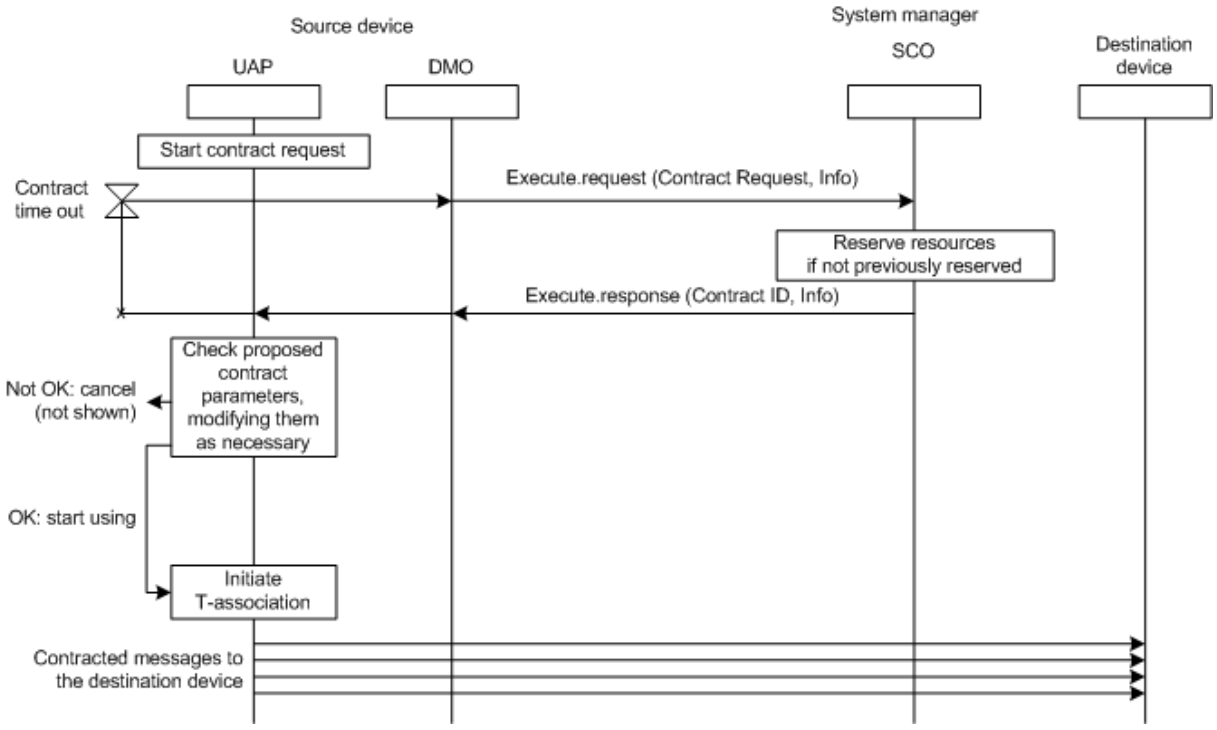


Figure 30 – Contract establishment example

6.3.11.2.9 Use of contract identifier

6.3.11.2.9.1 General

The contract ID shall be provided by the system manager to the source. The contract requesting application process shall retrieve the assigned contract ID and shall use it to send protocol data units down the protocol suite. As described previously, each layer of the source is configured for treating such upper layer PDUs that are accompanied by the contract ID, which is passed along as a DSAP control parameter.

Figure 31 illustrates how the contract ID shall be used as the data unit flows down the protocol suite of the source.

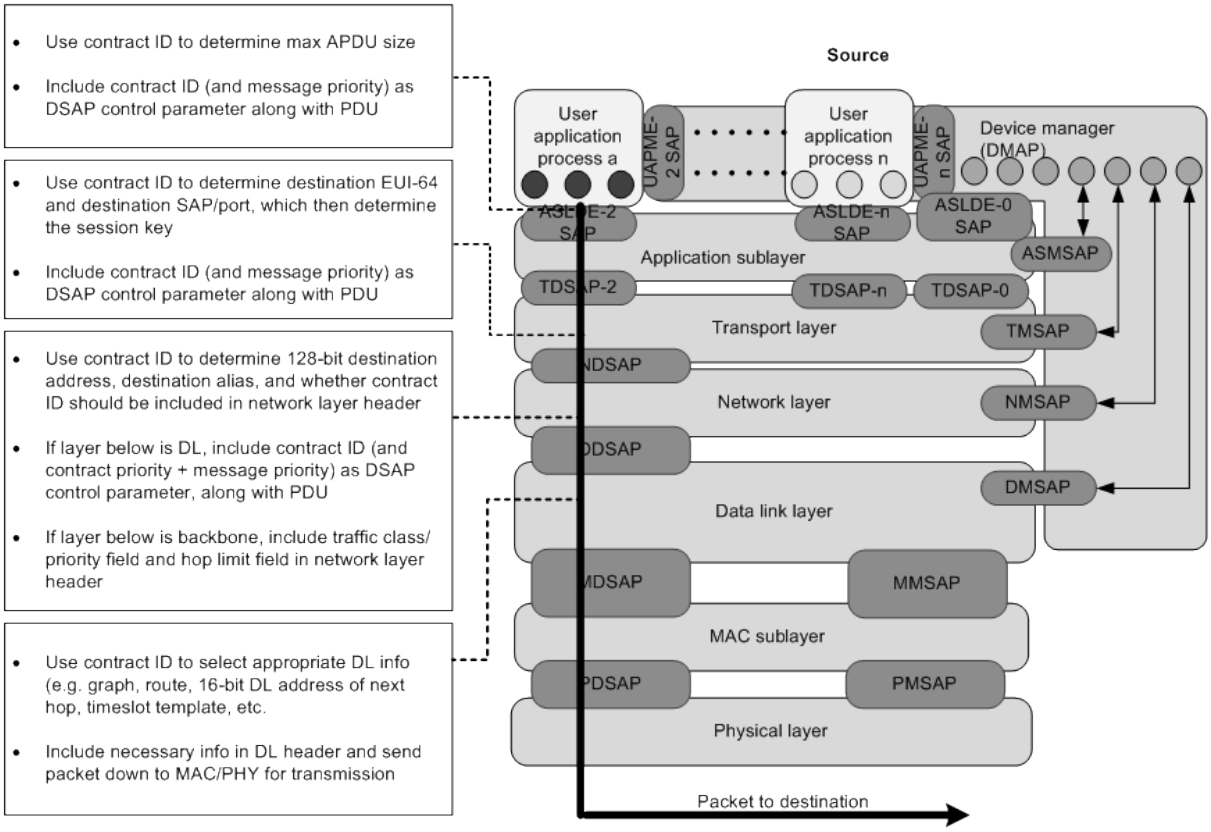


Figure 31 – Contract ID usage in source

6.3.11.2.9.2 Use of contract identifier in intermediate backbone routers

Inclusion of the contract ID in the network header of the NPDU by the source shall be configured by the system manager. If the communication path from the source to the destination goes through the backbone, then the system manager shall inform the source to include the contract ID in its network header. More details are provided in 10.5.3.

6.3.11.2.9.3 Relation between contracts and alerts

Access to the DMAP is restricted to the SMAP that resides in the system manager. In contradiction to this general principle, alert masters are allowed to access the ARMO object present in the DMAP. DMAP access by alert masters shall be limited to the ARMO, unless the alert master uses the DMAP-SMAP session established when the device joined the network. The ARMO in the DMAP shall transmit alerts that belong to the different alert categories to the respective alert masters which are described in 6.2.7.2. If these alert masters are different devices with their own unique IPv6Addresses, the ARMO shall have a separate contract with

each one to communicate the alerts. The ARMO in the device requests for these contracts from the system manager through the DMO in the device.

6.3.11.2.10 Contract termination, deactivation and reactivation

6.3.11.2.10.1 General

Contracts may be terminated when the communication need that established the contract has been satisfied. Contracts may also be terminated when either the source or the destination is no longer available.

When there is a contract termination, the SCO shall inform the DMO of the source, if the source is still available. The DMO in turn informs the application process that was using this contract.

When there is a contract termination, the SCO may also free up the network resources that were allocated for supporting the contract. In addition, security information, including T-keys between the source and the destination, may also be deleted by the security manager based on interactions with the system manager.

A contract may also be deactivated if the communication need is expected to be suspended for a period of time. The contract can be reactivated when the communication need resumes.

6.3.11.2.10.2 Contract termination when a device leaves the network or is no longer available

When the system manager determines that a device is no longer part of the network, it shall terminate all the contracts associated with that device and free up the network resources that were allocated for supporting those contracts. The system manager may use information from other devices in the neighborhood of this device to decide that this device is no longer part of the network. The system manager may read the dlmo.Neighbor attribute (described in 9.4.3.4) of these neighboring devices to make this decision.

When a device that has DMO attribute Non_Volatile_Memory_Capability = 1 loses network connectivity/power cycles or goes through a warm restart for any reason, it shall maintain all necessary information related to contracts as described in 6.3.9.4.2. So, this device can resume normal operation as soon as it re-establishes time synchronization with the network. The device is expected to re-establish time synchronization by listening for advertisements or by soliciting advertisements.

If the system manager terminated all the contracts of this device while the device was not part of the network, the device is expected to be unsuccessful in resuming normal operation and so is expected to execute a restartAsProvisioned cycle. This device shall retain all the information that was provided to it during the provisioning step before it first joined the network as well as all the constant and static information present in the UAPs.

When a device that has DMO attribute Non_Volatile_Memory_Capability = 0 loses network connectivity, cycles power, or undergoes a restartAsProvisioned cycle for any reason, it is expected to repeat the joining process by using the information that was provided to it during the provisioning step before it first joined the network. This device shall also retain all the constant and static information present in the UAPs.

The DMO of a device that is resetting to the factory default state or is undergoing a restartAsProvisioned cycle shall terminate all its contracts by using the method defined in Table 27 before resetting or restarting.

6.3.11.2.10.3 Contract termination when the T-key is terminated

The system manager may terminate a contract of a particular device if it is informed by the security manager that a corresponding T-key of that device has been terminated. Any contract of the device that uses the T-port corresponding to this particular T-key may be terminated.

6.3.11.2.10.4 Devices that can terminate, deactivate and reactivate contracts

Only the source or the system manager shall have the ability to terminate an existing contract of the source.

Only the source shall have the ability to deactivate and reactivate an existing contract of the source.

6.3.11.2.10.5 Contract termination, deactivation, and reactivation request and response arguments

If the source decides to terminate a contract, it shall send a contract termination request to the SCO. The SCO shall then send the response back to the source informing it that the contract has been terminated. The request shall be an Execute.Request to the SCO with the contract ID as one of the input arguments, and the response shall be an Execute.Response with status as the output argument. This is described in Table 32.

If the source decides to deactivate/reactivate a contract, it shall send a contract deactivation/reactivation request to the SCO. The SCO shall then send the response back to the source informing it that the contract has been deactivated/reactivated. The request shall be an Execute.Request to the SCO with the contract ID as one of the input arguments, and the response shall be an Execute.Response with status as the output argument. This is described in Table 32.

Table 32 – SCO method for contract termination, deactivation and reactivation

Standard object type name:System communication configuration object (SCO)				
Standard object type identifier: 102				
Method name	**Method ID**	**Method description**		
Contract_Termination _Deactivation_Reactivation	2	Method to terminate, deactivate or reactivate a contract		
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Contract ID	Unsigned16	ID of contract being terminated, deactivated or reactivated
	2	Operation	Unsigned8	Named values: 0: contract termination; 1: contract deactivation; 2: contract reactivation
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Error	Unsigned8	Named values: 0: success; > 0: failure

If the system manager decides to terminate a contract, it shall send a contract termination command with the contract ID to the DMO of the source. The DMO shall then return a response with the status.

The DMO method to notify an application that an existing contract has been terminated is described in Table 33.

Table 33 – DMO method to notify of contract termination

Standard object type name: Device management object (DMO)		
Standard object type identifier: 127		
Method name	**Method ID**	**Method description**
Contract_Terminated	1	Method to notify an application of the termination of an existing contract, as found in the Contracts_Table attribute in Table 10. This method uses the Delete_Row method template defined in Table J.5 with the following arguments: Attribute_ID: 26 (Contracts_Table) Index_1: 1 (Contract_ID)

6.3.11.2.10.6 Protocol suite configuration

When the SCO terminates a contract, in addition to informing the source about the termination, it may also free up the network resources that were allocated in the source, destination, and intermediate devices. Procedures similar to those used for protocol suite configuration during contract establishment (see 6.3.11.2.6) may be used by the SCO to free up these network resources.

The SCO informs the security manager through the PSMO about the contract termination. The security manager may decide to delete the T-key that has been assigned for the communication between the source and the destination. In this case, the security manager shall send T-key delete messages to the source and the destination through the PSMO.

6.3.11.2.10.7 Contract termination message sequence diagram

Figure 32 shows the message sequence chart for termination of a contract initiated by the system manager.

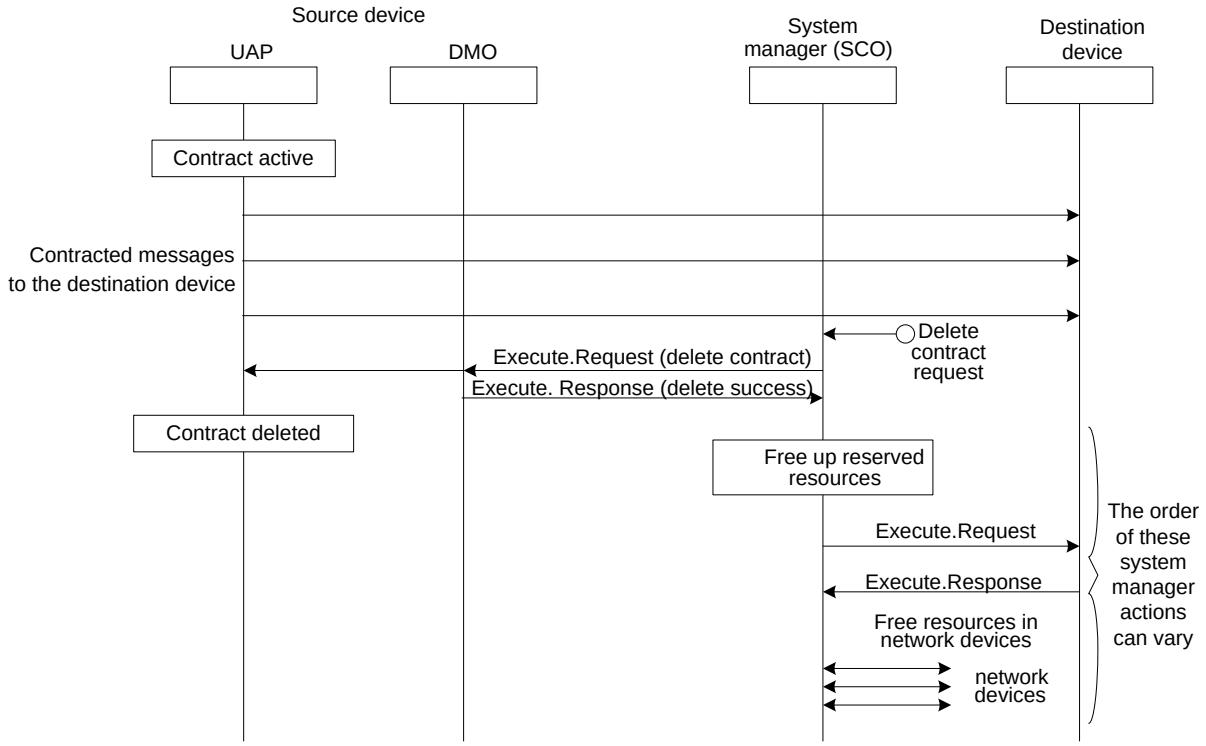


Figure 32 – Contract termination

6.3.11.2.11 Contract maintenance and modification

6.3.11.2.11.1 General

The SCO needs to maintain established contracts by ensuring that the allocated network resources are available under normal conditions. If the allocated network resources become unavailable, the SCO may choose to allocate alternate network resources in order to continue to maintain the established contract.

A contract may be modified if the communication need of the corresponding application (supported by that contract) changes. A contract may also be modified if the system manager decides to change the network resources allocated for the contract.

Contract modifications fall into two categories:

- modifications resulting in a reduction of the allocated network resources, and
- modifications resulting in a change or increase of allocated network resources.

For application-initiated contract modifications, these two categories follow slightly different steps.

Contract modifications that result in a reduction of the allocated network resources may go into immediate effect, i.e., the source may start using the protocol suite configuration of the modified contract as soon as it receives the response along with this configuration information from the SCO if this response indicates so in the Response_Code output argument.

Contract modifications that result in an increase or change of the allocated network resources shall not go into immediate effect, i.e., the source shall not start using the protocol suite configuration of the modified contract as soon as it receives the response along with this configuration information from the SCO. This is because the SCO still needs to increase or change the allocation of network resources. The response from the SCO shall include an

Activation_Time output argument that indicates to the source when it can start using the new protocol suite configuration. This results in a delayed effect.

6.3.11.2.11.2 Devices that can modify contracts

Only the source or the system manager shall have the ability to modify an existing contract of this source.

6.3.11.2.11.3 Contract modification request and response arguments

If the source decides to modify a contract, it shall send a contract modification request to the SCO. The SCO shall then send a response back to the source informing it that the contract has been modified. The request shall be an Execute.Request to the SCO, and the response shall be an Execute.Response message. The input and output arguments are provided in Table 27. The SCO may also communicate with relevant devices to allocate or de-allocate the necessary network resources.

If the system manager decides to modify a contract, it shall send a contract modification command to the DMO of the source by using the Modify_Contract method. The DMO shall then send a response back with the status. The SCO may also communicate with relevant devices to allocate or de-allocate the necessary network resources.

The DMO method to notify an application that an existing contract has been modified is described in Table 34.

Table 34 – DMO method to notify of contract modification

Standard object type name: Device management object (DMO)		
Standard object type identifier: 127		
Method name	**Method ID**	**Method description**
Contract_Modified	2	Method to notify an application of the modification of an existing contract, as found in the Contracts_Table attribute in Table 10. This method uses the Write_Row method template defined in Table J.3 with the following arguments: Attribute_ID: 26 (Contracts_Table) Index_1: 1 (Contract_ID)

6.3.11.2.11.4 Contract renewal

Contract renewal is equivalent to a simple contract modification, with only the Contract_Expiration_Time input argument being updated and all other input arguments being the same as those in the original contract request.

6.3.11.2.11.5 Protocol suite configuration

As part of contract modification, the SCO shall configure/re-configure the necessary devices in the network by providing the necessary protocol suite configurations to each of them. This shall include the re-configuration of the destination and the source. Procedures similar to the ones used for protocol suite configuration during contract establishment (see 6.3.11.2.6) may be used by the SCO for this purpose.

6.3.11.2.11.6 Contract modification message sequence diagram

Figure 33 shows the message sequence chart for modifying a contract with immediate effect.

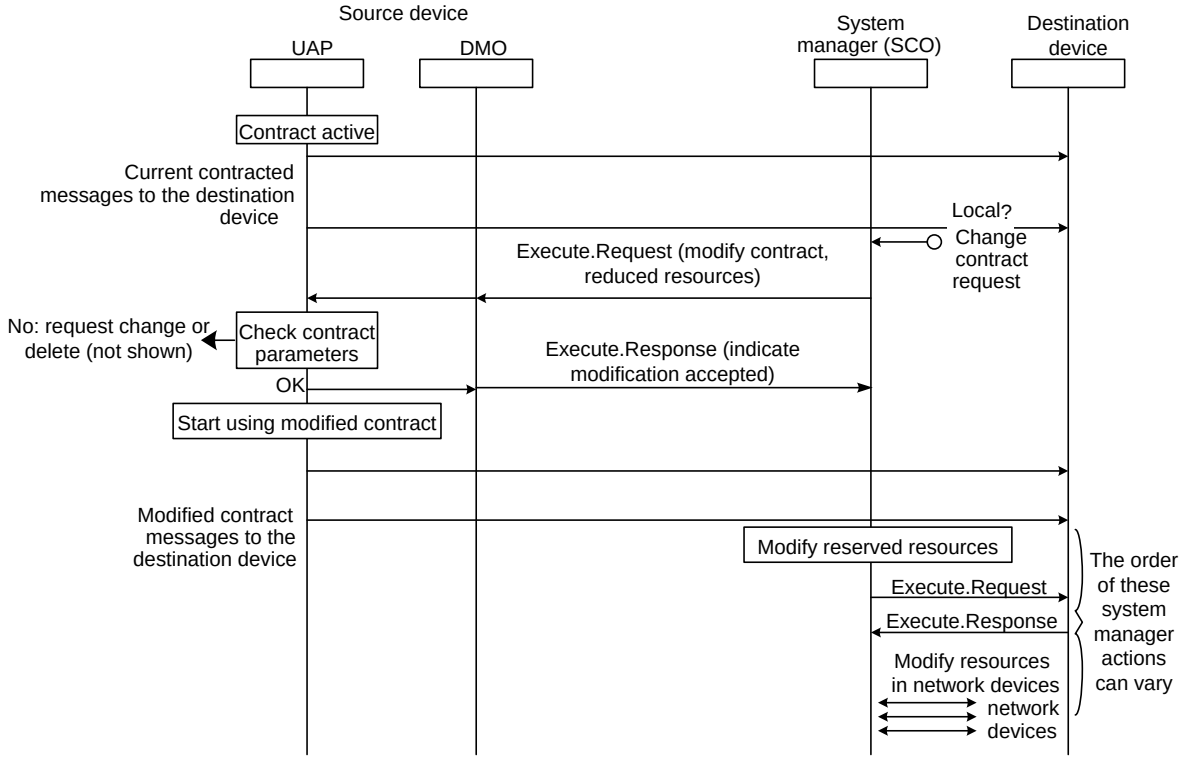


Figure 33 – Contract modification with immediate effect

6.3.11.2.11.7 Contract modification and T-key updates

T-key updates are not treated as contract modifications. Such key updates shall be sent from the security manager, through the proxy security management object (PSMO) in the system manager, to the relevant devices that have the corresponding session.

6.3.11.2.12 Contract failure scenarios

Table 27 contains output arguments for failure scenarios in which the system manager is not able to support the contract request. Such failures may occur if the requested communication service cannot be supported at all, if it cannot be supported due to a temporary condition, or if it cannot be supported unless the request is resent by the source with arguments negotiated down. In such cases, the system manager may choose to include output arguments in the response that provide some guidance to the source. These include Retry_Backoff_Time and Negotiation_Guidance.

6.3.12 Redundancy management

Although this standard incorporates features that provide for both simplex and fully-redundant wireless connection from field devices to a backbone plant network, the management of that redundancy is not specified in this standard.

The system manager is expected to be capable of configuring path redundancy in the D-subnet through the field routers. Field devices, including field routers, can be configured to communicate with redundant backbone routers.

Device-level redundancy that requires synchronization between the redundant devices to maintain state information is allowed, but is not specified in this standard.

6.3.13 System management protocols

Management-related communication between devices compliant with this standard and the system manager shall be accomplished via standard application sublayer messaging, as described in 12.12.

6.3.14 Management policies and policy administration

Management policies and policy administration are not specified by this standard. A default policy may be established to make all device information available to the system manager (with appropriate security). Overview information may be made available outside the network (e.g., the network is operating within nominal limits).

6.3.15 Operational interaction with plant operations or maintenance personnel

While the device implementing the system manager may have an interface that allows plant operations and maintenance personnel to observe and control the performance of the network and devices, this interface is neither mandatory nor is it specified by this standard.

7 Security

7.1 General

Clause 7 describes the security component functionality, its interface with the DLE and the TLE, and the protection of data in transit. It also describes the security manager role.

The primary focus of Clause 7 is to provide transmission security and related security aspects including the joining process, session establishment, key updates, and associated policies. This standard does not address other types of security, such as security of data-at-rest or physical device security.

The specific messages that are protected are single-hop (hop-by-hop) DPDUs, end-to-end transport TPDUs, and security management data structures when conveyed in APDUs. A steady-state data flow using DPDUs and TPDUs that may be protected is outlined in Figure 34. The TLE endpoints of a T-security association are defined by the endpoint devices as well as the end application.

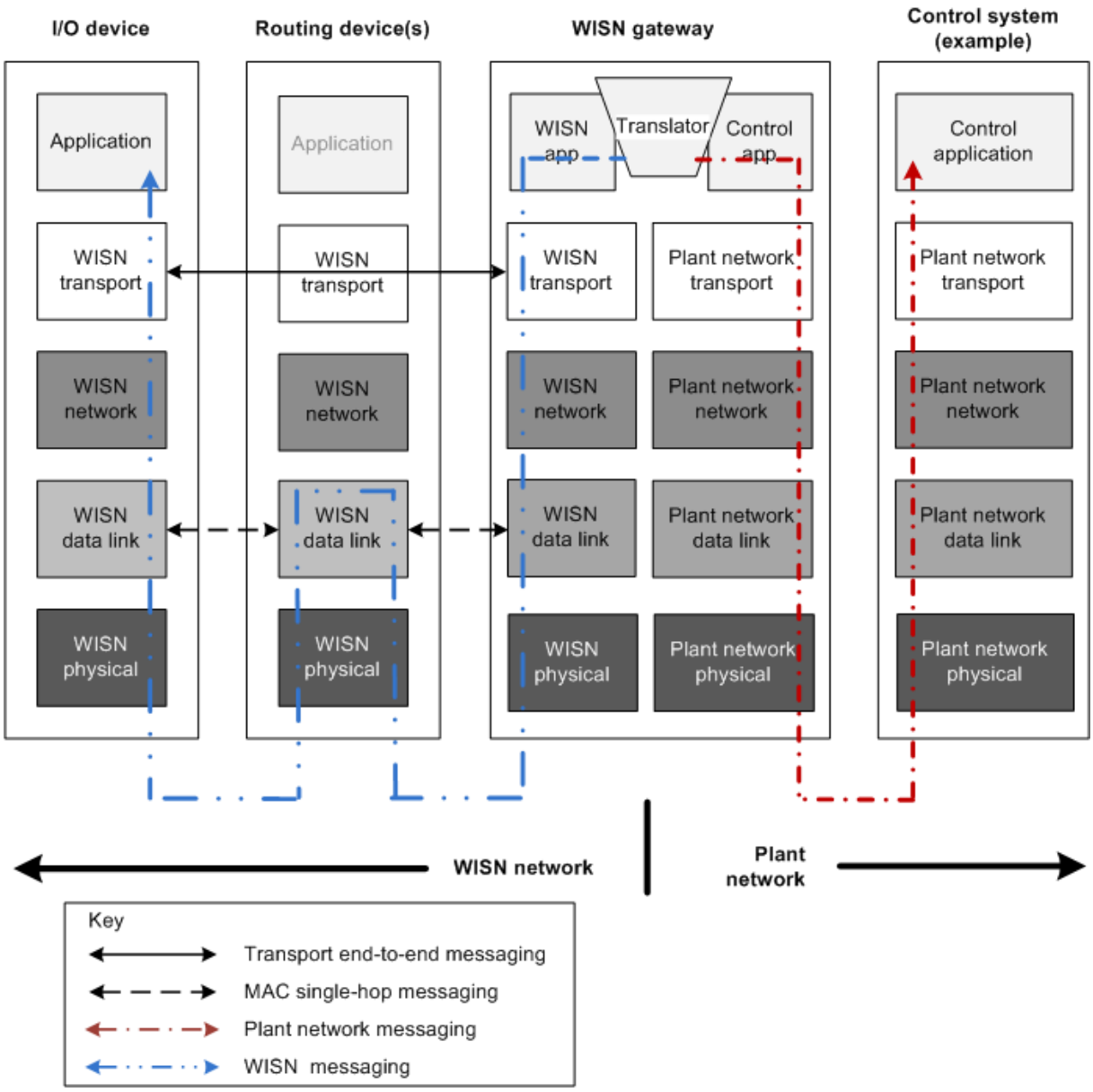


Figure 34 – Examples of DPDU and TPDU scope

7.2 Security services

7.2.1 Overview

The security services in this standard are selected by policy. The policy is distributed with each cryptographic material, permitting focused policy application. Since a single key is used at a time at the DL, except for a brief period of key switchover, the entire sub-network is subject to the same policies at the DL. The security manager controls the policies for all the cryptographic materials it generates.

Devices with appropriate credentials participate in secured communications with other such devices through the use of a shared-secret symmetric key that is used to authenticate and, by security configuration, to encrypt their messages to each other.

NOTE 1 Although authentication involves use of an encryption primitive, it does not result in confidentiality of the message contents; a separate encryption process (pass) is required for message content confidentiality.

The security services are applied at the bottom of the communication protocol stack, hop-by-hop at the DL, and at the top of the communication protocol stack, end-to-end at the TL.

Security management services are also used by the AL for the joining process, key distribution and session management. When secret keys are used, DL security defends against attackers which are outside the system and do not share system secrets, while TL security defends against attackers which may be on the network path between the source and the destination.

In both cases, a symmetric data key (also known as a T-key), shared among intended communicants, is used to add a cryptographically-hard, keyed, message integrity check (MIC) to the PDU and, when so specified, to provide confidentiality (via encryption) of the PDU's payload. Attackers that do not share the key cannot modify the message without a very high probability of detection and cannot decrypt any encrypted payload.

The security operation is based on a shared sense of time that usually is aligned with TAI time (see 5.6). The sending DLE and TLE authenticate to their receiving peers using the nominal TAI time of DPDU transmission and the approximate time of TPDU creation.

When three or more devices share a common secret key, source authentication is no longer guaranteed within that group because of the shared symmetric key. In this case, intra-group source authentication requires complex mechanisms; thus, authentication of the specific sending node (within the multicast group) is not addressed.

The primary security components of the provided services include:

- authorization of secure communications relationships between entities;
- message authenticity, ensuring that messages originate from an authorized member of a communications relationship and that they have not been modified while in transit between originator and receiver by an entity outside of the relationship;
- assurance that delivery timing and message reordering does not exceed anticipated bounds;
- data confidentiality that conceals the contents (other than size) among message payloads; and
- protection against malicious replay attack.

Various combinations of these services are provided to both a DLE and a TLE. Additionally, various cryptographic services are available for use by the DSMO for the joining process, session establishment and key update.

NOTE 2 Protection against compromise of the cryptographic boundaries inside the hardware of devices compliant with this standard is beyond the scope of this standard. Other publications, including ISO/IEC 15408 and ISO/IEC 19790 (similar to the [US] NIST FIPS 140 series), address those issues. Compliance decisions are left to those who evaluate devices.

7.2.2 Keys

7.2.2.1 General

Symmetric keys are used for data encryption and authentication (see 7.3.2.5, 7.3.2.6, 7.3.3.8, and 7.3.3.9). Asymmetric keys can be used for the joining process, see 7.4. Each key is limited in time and can be updated. Figure 35 shows the types of keys specified by this standard and their associated lifetimes, including an asymmetric-key security certificate (should one exist).

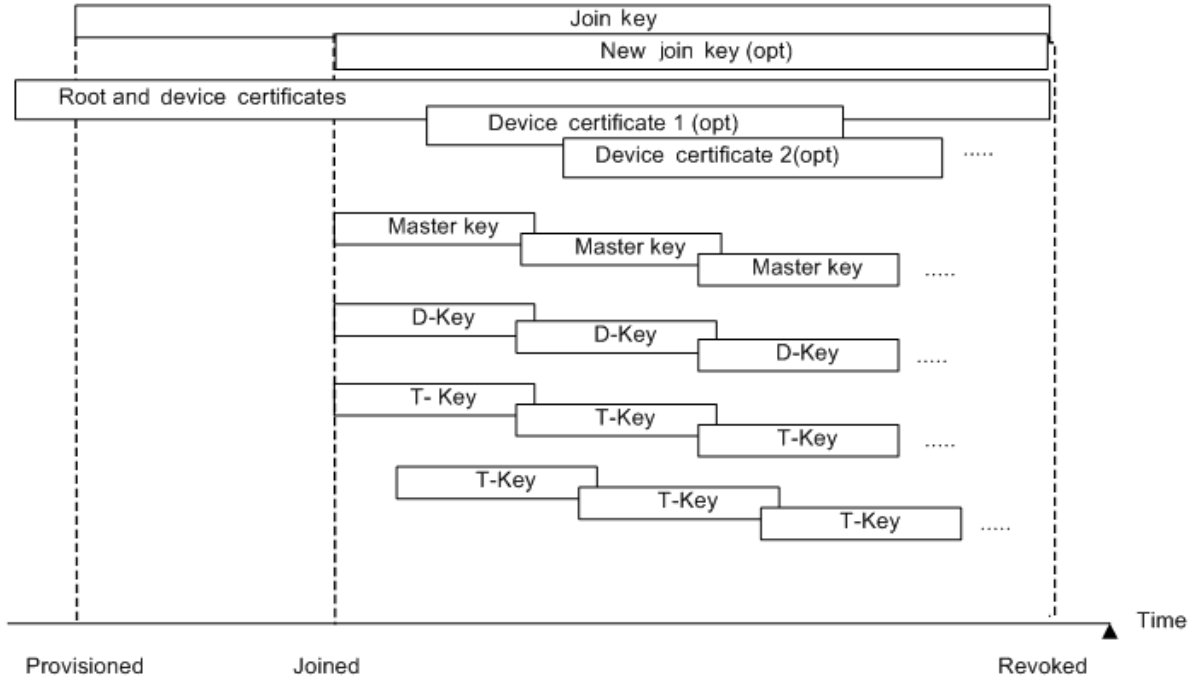


Figure 35 – Keys and associated lifetimes

7.2.2.2 Symmetric keys

All WISN symmetric keys shall be 128-bit values. The symmetric keys used include the following.

- Global key: a well-known key that cannot be used to guarantee any security properties and which never expires.
- K_open: a global key used as the join key in the provisioning step described in 13.3. The actual value for this key is 0x004F 0050 0045 004E 0000 0000 0000 0000, which is the representation of the null-terminated 16-octet Unicode string “OPEN(null)(null)(null)(null)”. The crypto key identifier for this key is 1.
- K_global: a global key used as the join key in the provisioning phase, and as the D-key in the joining phase. Use of this key in the provisioning phase is described in 13.3. The actual value for this key is 0x0049 0053 0041 0020 0031 0030 0030 0000, which is the representation of the null-terminated 16-octet Unicode string “ISA(space)100(null)“. The crypto key identifier for this key is 0.
- Join key (K_join): a key received at the conclusion of the provisioning step, is used to join a network for which the device was provisioned. The default value of the K_join key is the same as the default value of K_global.
- Master key: a key derived at the conclusion of a key agreement scheme, which is used as a KEK for communication between the security manager and the device, as well as a basis for deriving other keys. This key expires and needs to be updated periodically.
- D-key: a key used to encrypt/decrypt and/or authenticate DPDUs. This key expires and needs to be updated periodically.
- T-key: a key used to encrypt/decrypt and/or authenticate TPDUs. That key expires and needs to be periodically updated.

7.2.2.3 Asymmetric keys and certificates

Support of asymmeric cryptography is a device construction option.

All WISN asymmetric keys shall have a cryptographic strength of at least 128 –bits. The asymmetric keys used include:

- CA_root: The public key of the certificate authority that signed the device's asymmetric-key certificate. This key is commonly referred to as a root key; it is used in verifying the true identity of the device communicating the certificate, as well as some related keying information.
- Cert-A: The asymmetric-key certificate of device A, used to evidence the true identity of the device, as well as related keying information. It is used during execution of an authenticated asymmetric-key key establishment protocol.

The description of the asymmetric-key cryptographic material is provided in Clause H.3.

7.2.2.4 Key lifetime

7.2.2.4.1 General

Symmetric keys are limited by a lifetime and should be invalidated after the lifetime expires. To maintain security of ongoing communications, current DL- and TL-keys are updated at intervals during system operation. DL-keys are shared among all the communicating DLEs of a DL-subnet, whereas TL-keys are shared only between a pair (or potentially a larger set) of TLEs that have previously established a TL-communications relationship. In contrast, KEKs are shared only between a device and the system manager.

In this specification, the key lifetimes (and related information) are defined as follows:

ValidNotBefore:	TAI time at which a key will be enabled;
ValidNotAfter:	TAI time after which a key will become invalid;
SoftExpirationTime:	TAI time when a device should prepare for updating a key;
HardLifeSpan:	relative duration from ValidNotBefore to ValidNotAfter;
KeyExchangeMargin:	minimum time required to complete a key update cycle.

NOTE 1 Since the above are used herein as variables in formulae they use the typeface for variables.

The relationship of the above lifetime definitions is illustrated in Figure 36. The key update mechanism using those time definitions is described in 7.6.

The special value 0xFFFF FFFF is used to designate keys that never expire, which is used for global keys specified in 7.2.2.2. Thus any computation of the expiration time of a key shall replace a result value of 0xFFFF FFFF with 0x0000 0000. Similarly, any logic that determines whether a key has expired because the key's expiration time is in the near past shall determine that expiry has not occurred when that value for that expiration time is 0xFFFF FFFF.

NOTE 2 DL, TL, and KEK (device master) keys are safer if they do expire, since keys that do not expire increase the system's vulnerability to prolonged observation and attack.

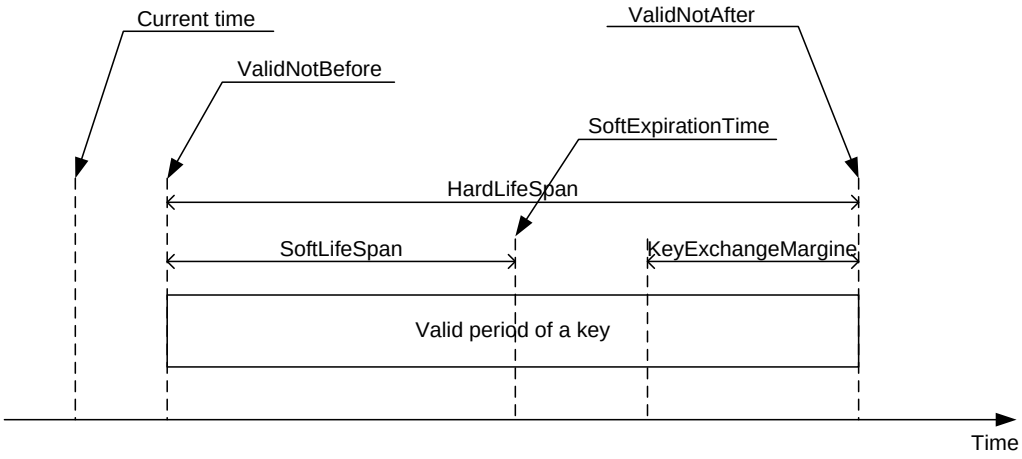


Figure 36 – Key lifetimes

NOTE 3 A key used after its hard lifetime can make communications vulnerable to replay attacks.

Asymmetric-key certificates should have a lifetime (ValidNotBefore and ValidNotAfter), as defined in 7.4.6.2.1.1.

KeyExchangeMargin can be used as a trigger for invoking the PSMO.Key_Update_Request() method to keep the continuous secure session. It is recommended that KeyExchangeMargin is set to “5 times DSMO.pduMaxAge” seconds, consisting of:

- 2 × DSMO.pduMaxAge seconds for a Security_New_Session() method round-trip communication;
- 2 × DSMO.pduMaxAge seconds for a New_Key() method round-trip communication; and
- another DSMO.pduMaxAge seconds for processing time.

7.2.2.4.2 Key lifetime expiration

7.2.2.4.2.1 SoftExpirationTime

When the SoftExpirationTime is past, the device owning the key prepares to get a new key from the security manager. The device may call the PSMO.Security_New_Session() method on the system manager to explicitly request a key, or it can wait to have its DSMO.New_Key() method called by the security manager. If the device wants to be certain about updating a key, it should call the PSMO.Security_New_Key() on the system manager method explicitly.

It is not necessary for the device to start the key update process immediately after SoftExpirationTime. The key update can be accomplished at any time up to ValidNotAfter. To keep the current secure session with a peer, a request for a new key can be issued at some point between SoftExpirationTime and ValidNotAfter. The device should call the PSMO.Security_New_Key() method before (HardExpirationTime – KeyExchangeMargin).

7.2.2.4.2.2 ValidNotAfter

The key shall not be used in active communication after ValidNotAfter and should be zeroized by all devices using the key. However, the key can be archived in a secure manner, depending on the system key archiving policy.

7.3 PDU security

7.3.1 General

7.3.1.1 Security level

The security level specifies the method to be applied to certain PDUs. The security level consists of a combination of the MIC size (0 bits, 32 bits, 64 bits or 128 bits) and whether the associated PDU payload is to be encrypted or not. Table 35 shows the security levels used in this specification and their corresponding security attributes.

Table 35 – Security levels

Security level value	Security attributes	Where usable
0	None	TPDU
1	MIC-32	Data DPDU, ACK/NAK DPDU, TPDU
2	MIC-64	Data DPDU, TPDU
3	MIC-128	TPDU
4	ENC-only	Never
5	ENC-MIC-32	TPDU, Data DPDU
6	ENC-MIC-64	TPDU, Data DPDUL
7	ENC-MIC-128	TPDU
NOTE 1 PhPDU size constraints and loss rates dictate the ACK/NAK DPDU restriction to MIC-32 and the Data DPDU restriction to MIC-32, ENC-MIC-32, MIC-64, or ENC-MIC-64. NOTE 2 ACK/NAK DPDUs do not contain a payload field to which the ENC operation could apply. NOTE 3 ENC-only is excluded because it is not possible to determine whether the eventual decryption is correct.		

7.3.1.2 Security control field

The security control field is part of each DL and TL security header. Its value specifies the presence of the key identifier and the security level to be applied to the PDU. The SecurityControl field octets shall conform to IEEE 802.15.4:2011, 7.4.1.

Table 36 shows the structure of the security control field.

Table 36 – Structure of the security control field

Octet	Bits							
	7	6	5	4	3	2	1	0
1	Reserved			Crypto key identifier mode		Security level		

The CryptoKeyIdentifierMode field encodes the size of the CryptoKeyIdentifier field that immediately follows the SecurityControl field in the PDU. If the key identifier mode is set to 0, the following CryptoKeyIdentifier field is elided.

The security level field shall consist of 3 bits as defined in IEEE 802.15.4:2011, Table 58, and summarized in Table 35 of this standard. The security level 0x04, corresponding to encryption only, shall never be used for a TPDU, or the first DPDU of a D-transaction, of this standard. The security level of 0x00, corresponding to no protection shall never be used for a DPDU in this standard.

NOTE ENC, encryption-only, does not provide any protection against an active attacker, because such an attacker is able to arbitrarily complement selected bits of any PDU in transit. Without a cryptographically-difficult-to-forge integrity field, there is no secure method for the recipient to detect such a change, and thus any active attacker can easily fabricate a malicious PDU.

7.3.2 DPDU security

7.3.2.1 General

The degree to which a device is permitted to participate in a D-subnet shall be determined by the system policy applied to credentials supplied by the device. Devices without credentials shall be permitted full, limited, or no participation beyond join attempts, as determined by the system policy for such devices.

All DPDUs include security fields and a cryptographically-strong DMIC. The details of the cryptographic building blocks are in Annex H. In non-secure mode, the key distributed might have traveled over an insecure channel. When a properly secured secret D-key is used, the following security services are always provided:

a) DPDU source-set authentication;

b) DPDU integrity; and

c) proof that the DPDU was received at the intended time, providing rejection of DPDUs
   - that were not sourced by a device within the network that shares an appropriate data key, or
   - that were not received within an acceptable time window relative to their nominal time of formation or transmission, or
   - that were previously received.

NOTE 1 Authorization is implied by the fact that the sending device has knowledge of a shared symmetric data key. When the key is not a shared secret, the authorization extends to all possible devices through the use of the global key. When the key is a shared secret, an inference is available that the sending device obtained the shared-secret key from a security manager, and that it would have obtained the key only if the security manager's authorization database permitted the resulting protected communications relationship. Such permission usually is based on the device's role. See Device_Role_Capability (standard object type identifier 127, attribute identifier 4, in Table 10) for a definition of the roles and their respective bitmap.

NOTE 2 The detection of reception at an inappropriate time renders ineffective attacks on the MAC message stream that are based on DPDU delay, reordering, or replay, since the transmission duration of each DPDU is greater than the 1 ms window within which such reordering would be undetectable.

NOTE 3 This service uses the sender's time of transmission, the receiver's time of reception, and the fact that MAC transmission and reception are highly concurrent to ensure that any DPDU received at an unintended time, including DPDU replay or DPDU stream reordering, will be detected and the anachronistic DPDU(s) rejected.

The amount of redundancy (i.e., DMIC size) that is used to provide DPDU integrity is selected by policy associated with the relevant data key.

The following additional DL security service is selectable by policy associated with the relevant D-key:

d) DPDU payload confidentiality (i.e., encryption).

This confidentiality service shall not be offered with the K_open and K_global keys specified in 7.2.2.2, because use of these keys with their well-known constant values renders confidentiality impossible.

7.3.2.2 DPDU structure

The structure of a DPDU is described in 9.3.1 and outlined in Figure 88 and Figure 37 in this standard, with the DSDU possibly encrypted and the MHR, DHR and DSDU protected by the DMIC.

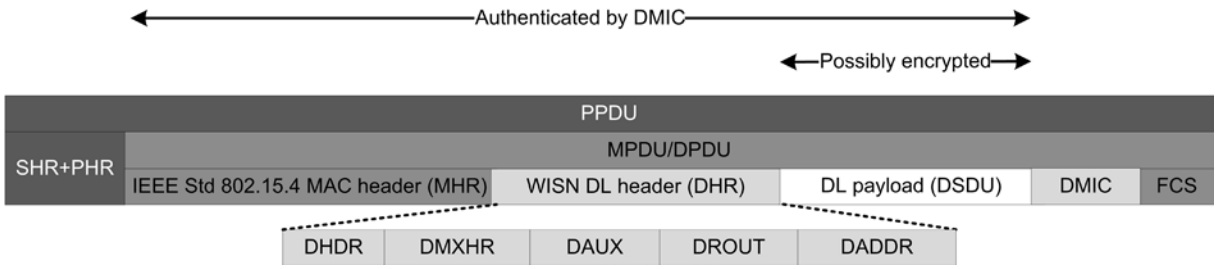


Figure 37 – DPDU structure

The complete DPDU from the start of the MHR to the end of the DSDU shall be protected by the DMIC. Information relevant to the DSC is the DL's MAC extension header (DMXHR) as outlined in Table 112, the 8-bit DPDU sequence number as outlined in Table 110, and the PhPDU's channel number (in the range 0..15).

7.3.2.3 DPDU headers

7.3.2.3.1 IEEE 802.15.4 MAC header

DPDU security is provided by the protocol stack defined in this standard, above the IEEE 802.15.4 MAC sublayer. The MAC header is defined in 9.3.3.2.

7.3.2.3.2 DL MAC extension header

The DMXHR outlined in Table 112 shall contain 2 fields used by the security layer. The first field shall contain the security control field as outlined in 7.3.1.2. The second field shall contain the Crypto Key Identifier as specified in IEEE 802.15.4:2011, 7.4.3. In the DMXHR, the Crypto Key Identifier shall never be elided with the Crypto Key Identifier Mode = 0.

The default value of the security level for the DL shall be set to 1 (MIC-32), corresponding to authentication only with a DMIC size of 32 bits.

For the DPDU processing steps, the following constraints shall be observed:

- DMIC sizes of 0 bits and 128 bits are prohibited, therefore prohibiting DPDU security levels of 0 (none), 3 (MIC-128), 4 (ENC-only) and 7 (ENC-MIC-128).

NOTE 1 MIC-64 provides adequate protection for Data DPDUs, given their small maximal size. That size constraint makes MIC-128 problematic, whereas the error rate of the underlying PhPDUs dictates that some MIC be used for additional DPDU integrity. A MIC also provides statistical protection against spoofing by an attacker that does not know the relevant symmetric encryption key.

NOTE 2 ENC-only is not useful because it is not possible on receipt to determine that the DPDU is received unchanged.

- ACK/NAK DPDUs shall use only 32-bit DMICs, regardless of the security level of the Data DPDU of a D-transaction.

NOTE 3 MIC-32 provides adequate protection for ACK/NAK DPDUs, given their minimal size and regulatory constraints on the duration of short control signaling (SCS), for which they qualify. ACK/NAK DPDUs carry no payload to which the DL's ENC (encryption) capabilities could be applied.

7.3.2.4 Interface between the DLE and DSC

7.3.2.4.1 General

Figure 38 summarizes the relationship between the DLE and DSC for DPDU transactions. This flow covers the normal case where a DPDU is transmitted and acknowledged, and no errors occur. For more detail, see the documentation for the corresponding DSAPs in 7.3.2.4.2, 7.3.2.4.3, 7.3.2.4.4, 7.3.2.4.5, 7.3.2.4.6, 7.3.2.4.7, 7.3.2.4.8, and 7.3.2.4.9.

All interfaces between the DLE and DSC are internal interfaces within the DLE, and thus are unobservable. Therefore they are not subject to standardization.

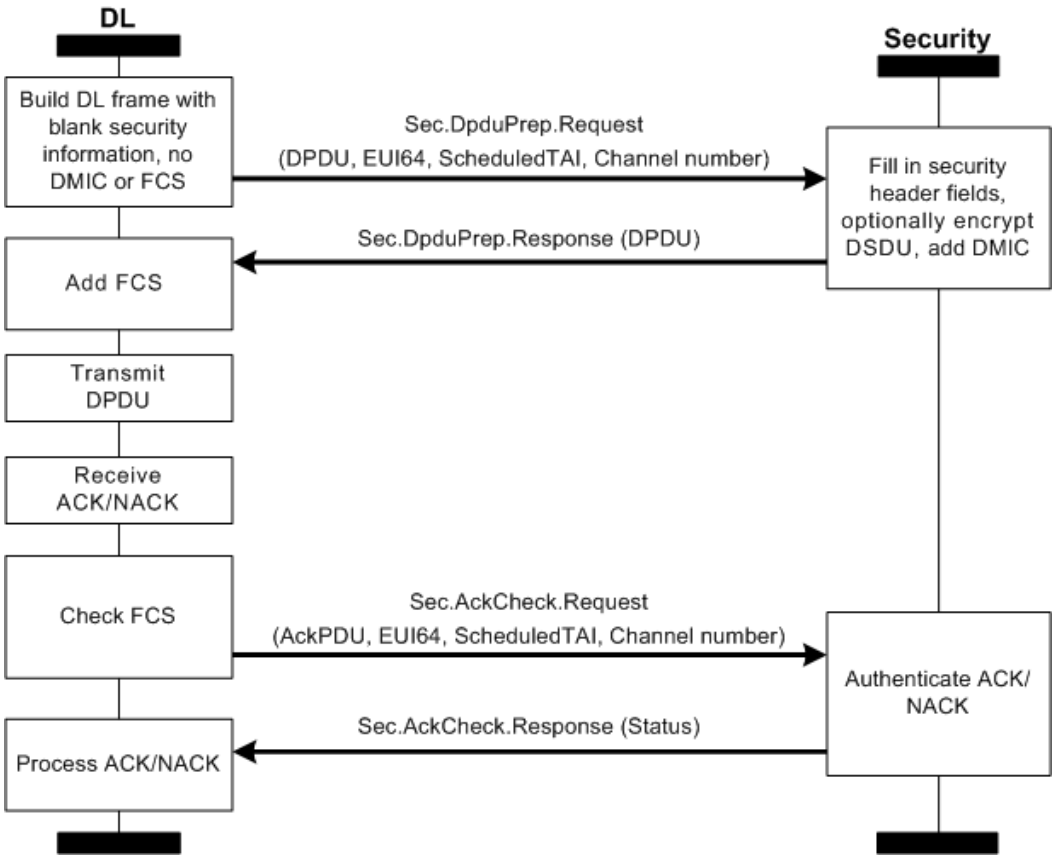


Figure 38 – DLE and DLS processing for a D-transaction initiator

The DLE assembles the DPDU to be protected. By documentation convention, security fields in the DPDU's header are populated by the DSC.

Certain DPDU security information is provided by the DLE to the DSC:

- the scheduled TAI time of the timeslot, used in the nonce to detect delayed and replayed DPDUs;

NOTE 1 The scheduled TAI time passed to the DSC is the scheduled TAI start time of the timeslot to which the D-transaction is assigned. The DSC truncates this time to 2^{-10} s (approximately 1 ms).

NOTE 2 There is one scenario under this standard where a single device might initiate multiple transmissions that all have the same scheduled timeslot start time. In that case a device (usually a backbone router) operates on multiple channels simultaneously, using synchronized timeslot templates in such a way that it can use either a single shared antenna or multiple closely-spaced antennas, phased in such a way that transmissions on one or more channels do not disrupt reception on other channels. While such operation is not explicitly described by this standard it is also intentionally not prohibited. Support for such concurrent operation gives rise to the following two nonce components that are included in the DPDU's nonce construction.

- the channel number of each DPDU, used in the nonce to detect DPDUs constructed for use in one channel that are replayed within the same timeslot in another channel;

NOTE 3 The channel number uses the channel-numbering convention of this standard, where the numbers 0..15 correspond to IEEE 802.15.4 channels 11..26 respectively.

- the one-octet sequence number found in the MAC header, used in the nonce to differentiate between the Data DPDU of a D-transaction and any ACK/NAK DPDUs that might be generated in timeslots with the same scheduled TAI start time;

NOTE 4 The low-order bits of the MHR sequence number octet encode the DPDU's zero-origin position in the D-transaction: 0 for the Data DPDU, 1 for the first ACK/NAK DPDU, 2 for a second ACK/NAK DPDU, etc.

- the DSC needs the EUI64Address of the destination device in order to process its ACK/NAK DPDU;

NOTE 5 When known to the DLE, this D-address is retrieved directly from the dlmo.Neighbor table.

- when a unicast destination's EUI64Address is not known to the DLE, the EUI64Address-requested indicator in the DHDR frame control octet (Table 111) shall be set (to 1), which triggers the destination to return its EUI64Address in the ACK/NAK DPDU.

NOTE 6 The DSC uses the DSDU size to encrypt only the DSDU and not the DPDU header, whereas the DMIC protects the entire DPDU. This detail is not shown in Figure 38 or Figure 39.

The DLE retains a copy of the outgoing DMIC, to be used subsequently to unambiguously connect the reply ACK/NAK DPDUs to the Data DPDU of the D-transaction. The DLE then appends an IEEE 802.15.4 FCS to the DPDU and transmits it without undue delay.

When the DLE receives an ACK/NAK DPDU, it requests the DSC to authenticate the DPDU. Certain DPDU security information is provided by the DLE to the DSC:

- Each ACK/NAK DPDU shall echo the D-transaction's initial DPDU's DMIC as a virtual field (see Table 117) in the computation of its D-MIC. The full ACK/NAK DPDU, including this virtual field, is reconstructed by the DLE before it is checked by the DSC.
- The EUI64Address of the ACK/NAK DPDU's originator is either looked up or provided within the ACK/NAK DPDU itself.
- The scheduled TAI time of the start of the D-transaction's timeslot, which is usually the same as the TAI start-of-timeslot time used by the D-transaction initiator. However, when slow-channel-hopping is used, the ACK/NAK DPDU may include a timeslot offset (see 9.3.4), in which case the nonce formed to check the ACK/NAK DPDU shall use the scheduled TAI start time of the timeslot referenced by the timeslot offset; that is, the scheduled timeslot of the acknowledging DLE.
- The channel number for sending the ACK/NAK DPDU is provided to the DSC.
- The MHR sequence number is provided to the DSC in the same manner that it is provided for the Data DPDU of the D-transaction.

Figure 39 illustrates the relationship between a DLE and its DSC for D-transactions in which the DLE is a recipient of or respondent to the D-transaction's Data DPDU.

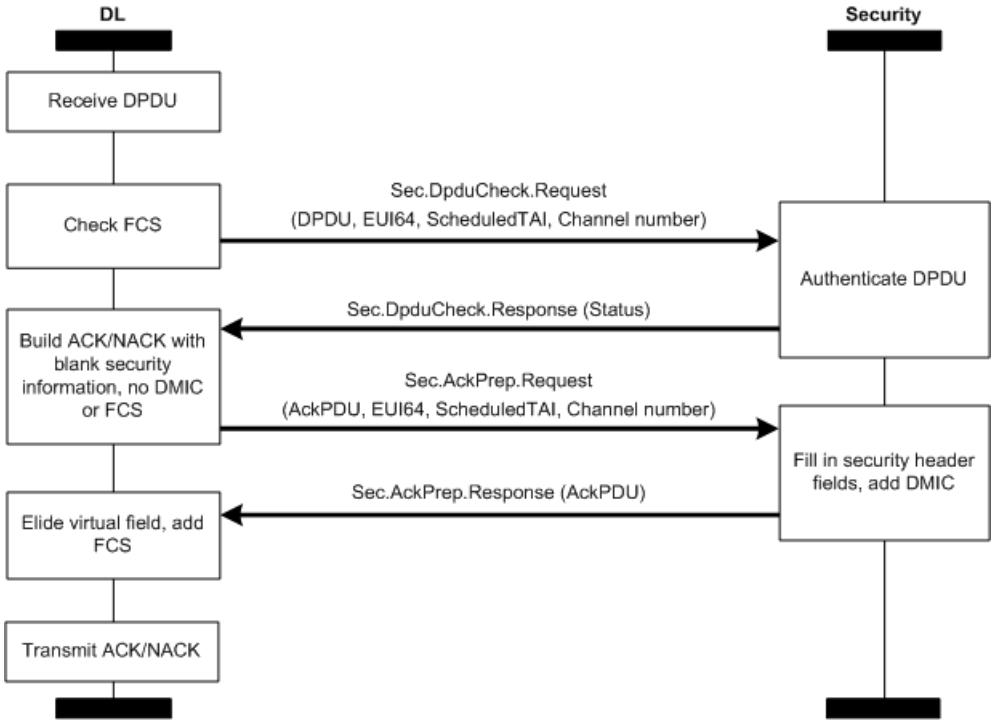


Figure 39 – Received DPDUs – DLE and DSC

When receiving a DPDU, the DLE sends a request to the DSE to authenticate the DPDU and, if necessary, decrypt the DPDU payload. The scheduled TAI time and the channel number are included with this request. The EUI64Address of the DPDU's source needs to be known by the DLE a priori, except in the case of a join request where it is carried in the DPDU header as a source address. The DSC normally responds with a positive authentication.

The DLE constructs the ACK/NAK DPDU. The ACK/NAK DPDU shall use the same scheduled TAI time as the received Data DPDU of the D-transaction, except when a slow-channel-hopping-offset correction is provided in the ACK/NAK DPDU as discussed above. The ACK/NAK DPDU shall also echo the DMIC of the initial received DPDU of the D-transaction as a virtual field, as shown in Table 117. The DSC then secures the ACK/NAK DPDU, including the DPDU's DMIC as a virtual field. The DLE elides the virtual field, appends an IEEE 802.15.4 FCS, and transmits the ACK/NAK DPDU.

When a DLE's local time sense is corrected by an ACK DPDU, such that its time is reset to an earlier timeslot, there shall be a forced pause in service, equal to the magnitude of the timeslot correction plus at least one timeslot.

7.3.2.4.2 Sec.DpduPrep.Request

7.3.2.4.2.1 General

Sec.DpduPrep.Request instructs the DSC to protect a DL protocol data unit as appropriate.

7.3.2.4.2.2 Semantics of the service primitive

The semantics of Sec.DpduPrep.Request are as follows:

```
Sec.DpduPrep.Request    (
    DPDU,
    EUI64,
    ScheduledTAI,
```

ChannelNumber,

AckHandle)

Table 37 specifies the elements for the Sec.DpduPrep.Request.

Table 37 – Sec.DpduPrep.Request elements

Element name	Element identifier	Element scalar type
DPDU (the DPDU to be transmitted)	1	Type: OctetString
EUI64 (the EUI64Address of the sending device)	2	Type: EUI64Address
ScheduledTAI (32-bits of the start time of the slot truncated to a 2^{-10} s resolution)	3	Type: Unsigned32
ChannelNumber (the channel number used in the transmitted DPDU)	4	Type: Unsigned8 Valid range: 0..15
AckHandle (abstraction that connects each invocation of Sec.DpduPrep.Request with the subsequent callback by Sec.DpduPrep.Response)	5	Type: Abstract

The DSC provides the DLE with the appropriate security control (octet 1) and the Crypto Key Identifier (octet 2) obtained from the KeyDescriptor for the current D-key, to be used in the DPDU's DMXHR subheader, the format of which is described in 9.3.3.4. See 7.3.2.5 on selecting the proper D-key.

The DSC populates the DMIC field as specified by the policy of the selected D-key.

7.3.2.4.2.3 Appropriate usage

The DLE invokes the Sec.DpduPrep.Request primitive to add security protection to a DPDU before it is transmitted.

7.3.2.4.2.4 Effect on receipt

On receipt of the Sec.DpduPrep.Request primitive, the DSC starts the appropriate DPDU processing steps to protect the DPDU as dictated by policy.

7.3.2.4.3 Sec.DpduPrep.Response

7.3.2.4.3.1 General

Sec.DpduPrep.Response reports the result of a Sec.DpduPrep.Request.

7.3.2.4.3.2 Semantics

The semantics of Sec.DpduPrep.Response are as follows:

Sec.DpduPrep.Response (

DPDU,

Status,

AckHandle)

Table 38 specifies the elements for Sec.DpduPrep.Response.

Table 38 – Sec.DpduPrep.Response elements

Element name	Element identifier	Element scalar type
DPDU	1	Type: OctetString
Status (the result of a Sec.DpduPrep.Request primitive)	2	Type: Unsigned Named values: 0: success; > 0: failure
AckHandle (abstraction that connects each invocation of Sec.DpduPrep.Request with the subsequent callback by Sec.DpduPrep.Response)	3	Type: Abstract

7.3.2.4.3.3 When generated

The DSC generates Sec.DpduPrep.Response in response to a Sec.DpduPrep.Request. The Sec.DpduPrep.Response returns a status value that indicates either SUCCESS and the unsecured DPDU or the appropriate error code.

7.3.2.4.3.4 Appropriate usage

On receipt of Sec.DpduPrep.Response, the DL is notified of the result of request to protect an outgoing DPDU.

7.3.2.4.4 Sec.DAckCheck.Request

7.3.2.4.4.1 General

Sec.DAckCheck.Request instructs the DSC to verify an incoming ACK/NAK DPDU.

7.3.2.4.4.2 Semantics of the service primitive

The semantics of Sec.DAckCheck.Request are as follows:

```
Sec.DAckCheck.Request     (
    AckPDU,
    EUI64,
    ScheduledTAI,
    ChannelNumber,
    AckHandle)
```

Table 39 specifies the elements for the Sec.DAckCheck.Request.

Table 39 – Sec.DAckCheck.Request elements

Element name	Element identifier	Element scalar type
AckPDU (the AckPDU to be verified)	1	Type: OctetString
EUI64 (the EUI64Address of the acknowledging device)	2	Type: EUI64Address
ScheduledTAI (32-bits of the start time of the slot truncated to a 2^{-10} s resolution)	3	Type: Unsigned32
ChannelNumber (the channel number used to receive the incoming ACK/NAK DPDU)	4	Type: Unsigned Valid range: 0..15
AckHandle (abstraction that connects each invocation of Sec.DAckCheck.Request with the subsequent callback by Sec.DAckCheck.Response)	5	Type: Abstract

The DSC verifies that the DHR of the ACK/NAK DPDU has employed the DMIC mode (see Table 118) specified by the current D-key policy. The D-key used in authenticating the ACK/NAK DPDU is the same as that used for the Data DPDU of the D-transaction.

The DSC verifies the DMIC field as dictated by the DPDU processing steps and current policies.

7.3.2.4.4.3 Appropriate usage

The DLE invokes the Sec.DAckCheck.Request primitive to verify an ACK/NAK DPDU after its reception.

7.3.2.4.4.4 Effect on receipt

On receipt of the Sec.DAckCheck.Request primitive, the DSC performs the appropriate DPDU processing steps to verify the received ACK/NAK DPDU as specified in 7.3.2.6.

7.3.2.4.5 Sec.DAckCheck.Response

7.3.2.4.5.1 General

Sec.DAckCheck.Response reports the result of a Sec.DAckCheck.Request.

7.3.2.4.5.2 Semantics

The semantics of Sec.DAckCheck.Response are as follows:

```
Sec.DInitialCheck.Response (
    AckPDU,
    Status,
    AckHandle)
```

Table 40 specifies the elements for Sec.DAckCheck.Response.

Table 40 – Sec.DAckCheck.Response elements

Element name	Element identifier	Element scalar type
AckPDU	1	Type: OctetString
Status (the result of a Sec.DAckPrep.Request primitive)	2	Type: Unsigned Named values: 0: success; > 0: failure
AckHandle (abstraction that connects each invocation of Sec.DAckCheck.Request with the subsequent callback by Sec.DAckCheck.Response)	3	Type: Abstract

7.3.2.4.5.3 When generated

The DSC generates Sec.DAckCheck.Response in response to a Sec.DAckCheck.Request. The Sec.DAckCheck.Response returns a status value that indicates either SUCCESS or the appropriate error code.

7.3.2.4.5.4 Appropriate usage

On receipt of Sec.DAckCheck.Response, the DL is notified of the result of verifying and possibly decrypting an incoming DPDU.

7.3.2.4.6 Sec.DInitialCheck.Request

7.3.2.4.6.1 General

Sec.DInitialCheck.Request instructs the DSC to verify and possibly decrypt an incoming DL protocol data unit as appropriate.

7.3.2.4.6.2 Semantics of the service primitive

The semantics of Sec.DInitialCheck.Request are as follows:

```
Sec.DInitialCheck.Request   (
    DPDU,
    EUI64,
    ScheduledTAI,
    ChannelNumber,
    AckHandle)
```

Table 41 specifies the elements for the Sec.DInitialCheck.Request.

Table 41 – Sec.DInitialCheck.Request elements

Element name	Element identifier	Element scalar type
DPDU (the DPDU to be verified and possibly decrypted)	1	Type: OctetString
EUI64 (the EUI64Address of the sending device)	2	Type: EUI64Address
ScheduledTAI (32-bits of the start time of the slot truncated to a 2^{-10} s resolution)	3	Type: Unsigned32
ChannelNumber (the channel number used to receive the incoming DPDU)	4	Type: Unsigned Valid range: 0..15
AckHandle (abstraction that connects each invocation of Sec.DInitialCheck.Request with the subsequent callback by Sec.DInitialCheck.Response)	5	Type: Abstract

The DSC verifies that the DMXHR of the DPDU has the appropriate security control (octet 1) by comparing it to the current policy. The Crypto Key Identifier (octet 2) is used to retrieve the correct key material. See 7.3.2.6.

The DSC verifies the DMIC field as dictated by the current policies.

7.3.2.4.6.3 Appropriate usage

The DL invokes the Sec.DInitialCheck.Request primitive to verify and possibly decrypt a DPDU before it is transmitted.

7.3.2.4.6.4 Effect on receipt

On receipt of the Sec.DInitialCheck.Request primitive, the DSC starts the appropriate PDU processing steps to verify the incoming DPDU as dictated by the incoming PDU processing steps in 7.3.2.6.

7.3.2.4.7 Sec.DInitialCheck.Response

7.3.2.4.7.1 General

Sec.DInitialCheck.Response reports the result of a Sec.DInitialCheck.Request.

7.3.2.4.7.2 Semantics

The semantics of Sec.DInitialCheck.Response are as follows:

```
Sec.DInitialCheck.Response (
                        DPDU,
                        Status,
                        AckHandle)
```

Table 42 specifies the elements for Sec.DInitialCheck.Response.

Table 42 – Sec.DInitialCheck.Response elements

Element name	Element identifier	Element scalar type
DPDU	1	Type: OctetString
Status (the result of a Sec.DpduPrep.Request primitive)	2	Type: Unsigned Named values: 0: success; > 0: failure
AckHandle (abstraction that connects each invocation of Sec.DInitialCheck.Request with the subsequent callback by Sec.DInitialCheck.Response)	3	Type: Abstract

7.3.2.4.7.3 When generated

The DSC generates Sec.DInitialCheck.Response in response to a Sec.DInitialCheck.Request. The Sec.DInitialCheck.Response returns a status value that indicates either SUCCESS or the appropriate error code.

7.3.2.4.7.4 Appropriate usage

On receipt of Sec.DInitialCheck.Response, the DL is notified of the result of verifying and possibly decrypting an incoming DPDU.

7.3.2.4.8 Sec.DAckPrep.Request

7.3.2.4.8.1 General

Sec.DAckPrep.Request instructs the DSC to protect an ACK/NAK DPDU as appropriate.

7.3.2.4.8.2 Semantics of the service primitive

The semantics of Sec.DAckPrep.Request are as follows:

```
Sec.DAckPrep.Request      (
    AckPDU,
    EUI64,
    ScheduledTAI,
    ChannelNumber,
    AckHandle)
```

Table 43 specifies the elements for the Sec.DAckPrep.Request.

Table 43 – Sec.DAckPrep.Request elements

Element name	Element identifier	Element scalar type
AckPDU (includes the virtual header)	1	Type: OctetString
EUI64 (the EUI64Address of the acknowledging device)	2	Type: EUI64Address
ScheduledTAI (32-bits of the start time of the slot truncated to a 2^{-10} s resolution)	3	Type: Unsigned32
ChannelNumber (the channel number used to transmit the ACK/NAK DPDU)	4	Type: Unsigned8 Valid range: 0..15
AckHandle (abstraction that connects each invocation of Sec.DAckPrep.Request with the subsequent callback by Sec.DAckPrep.Response)	5	Type: Abstract

The DSC populates the ACK/NAK DPDU with the appropriate security control (octet 1) as described in Table 118. In the case where multiple D-keys are currently valid, the key used to authenticate the ACK/NAK DPDU is the same one used as the corresponding DPDU for this ACK/NAK DPDU.

The DSC populates the DMIC field as dictated by the current policies. Note that the DMIC field in an ACK/NAK DPDU is always 32 bits.

7.3.2.4.8.3 Appropriate usage

The DL invokes the Sec.DAckPrep.Request primitive to protect an ACK/NAK DPDU before it is transmitted.

7.3.2.4.8.4 Effect on receipt

On receipt of the Sec.DAckPrep.Request primitive, the DSC starts the appropriate PDU processing steps to protect the ACK/NAK DPDU as dictated by policy. Note that the ACK/NAK DPDU is only authenticated and never encrypted.

7.3.2.4.9 Sec.DAckPrep.Response

7.3.2.4.9.1 General

Sec.DAckPrep.Response reports the result of a Sec.DAckPrep.Request.

7.3.2.4.9.2 Semantics

The semantics of Sec.DAckPrep.Response are as follows:

```
Sec.DAckPrep.Response     (
      AckPDU,
      Status,
      AckHandle)
```

Table 44 specifies the elements for Sec.DAckPrep.Response.

Table 44 – Sec.DAckPrep.Response elements

Element name	Element identifier	Element scalar type
AckPDU	1	Type: OctetString
Status (the result of a Sec.DAckPrep.Request primitive)	2	Type: Unsigned Named values: 0: success; > 0: failure
AckHandle (abstraction that connects each invocation of Sec.DAckPrep.Request with the subsequent callback by Sec.DAckPrep.Response)	3	Type: Abstract

7.3.2.4.9.3 When generated

The DSC generates Sec.DAckPrep.Response in response to a Sec.DAckPrep.Request. Sec.DAckPrep.Response returns a status value that indicates either SUCCESS or the appropriate error code.

7.3.2.4.9.4 Appropriate usage

On receipt of Sec.DAckPrep.Response, the DL is notified of the result of the request to verify an incoming AckPDU.

7.3.2.4.10 Nonce construction for DPDUs

This standard uses a different DPDU nonce construction from that of IEEE 802.15.4. A 13-octet nonce is required for the CCM* engine. The nonce shall be constructed as the concatenation from the first (leftmost) to the last (rightmost) octets of data fields as shown in Table 45, wherein:

- the EUI64Address shall be used as an array of 8 octets (in MSB convention) in the same manner as the source address of the CCM* nonce in IEEE 802.15.4:2011, 7.3.2;
- the TAI time shall be the least significant 32 bits of the TAI time in units of 2^{-10} s as described in Table 46;
- the last octet shall be constructed as follows:
  - Bit 7 shall be zero, thereby reserving the value 0xFF for the transport layer (see Table 57).
  - Bits 6..3 (4 bits) shall indicate the radio channel of transmission, in a range of 0..15, corresponding to IEEE 802.15.4 channel numbers 11..26, in the same order.
  - Bits 2..0 shall be copied from the corresponding low-order 3 bits of the MHR's sequence number.

Table 45 – Structure of the WISN DPDU nonce

Octet	Bits							
	7	6	5	4	3	2	1	0
1	EUI64Address of DPDU originator							
…								
8								
9	Least significant 32 bits of TAI time of nominal slot start (in units of 2^{-10} s)							
…								
12								
13	Reserved = 0	Channel number (0..15)				Low-order 3 bits of MHR sequence number		

The TAI time used shall be a 32-bit truncated fixed-point fractional representation of TAI time at a granularity of 2^{-10} s and a span of 2^{22} s. With this representation, there will be over 48,5 days before the same value of TAI time recurs. Thus, the maximum lifetime of a D-key shall be 48,5 days before a new D-key needs to be deployed. The TAI time for this operation shall be that maintained by the DLE.

NOTE 1 It is important that the value of the 32-bit representation of TAI time does not recur within the lifetime of any relevant secret symmetric key, to avoid a potential nonce collision resulting in an identical keystream.

The representation in the D-nonce of this truncated 32-bit TAI time, specified to 2^{-10} s, is described in Table 46.

Table 46 – Structure of the 32-bit truncated TAI time used in the D-nonce

Octet	Bits							
	7	6	5	4	3	2	1	0
1	Truncated TAI time (bits with weight $2^{21}..2^{14}$ s)							
2	Truncated TAI time (bits with weight $2^{13}..2^{6}$ s)							
3	Truncated TAI time (bits with weight $2^{5}..2^{-2}$ s)							
4	Truncated TAI time (bits with weight $2^{-3}..2^{-10}$ s)							

The lower order 3 bits of the MHR sequence number, together with the channel number, are used to construct the last octet of a D-nonce. The sending DLE shall ensure that the MHR sequence number bits used in the D-nonce are unique among all those it generates within the same 2^{-10} s interval for the same channel and same D-key (see 9.3.3.2 and 9.3.4). The value of 0xFF shall not be used for the MHR. Because this D-nonce has at most eight distinct values for a given channel and 2^{-10} s interval, a DLE shall not transmit more than eight DPDUs per 2^{-10} s on the same channel using the same D-key.

NOTE 2 Inclusion of the channel number in the D-nonce provides support for devices that operate concurrently on multiple channels.

NOTE 3 The construction of the MHR sequence number is described in 9.3.3.2.

7.3.2.5 Processing of a DPDU to be transmitted

The inputs to the DPDU security procedure are:

- the DPDU to be secured;
- the EUI64Address of the source DLE;
- the nominal TAI start time of the timeslot being used for the D-transaction;

- the MHR sequence number octet; and
- the channel number (0..15) to be used for the D-transaction.

The outputs from this procedure are:

- the status of the procedure; and
- if this status is success, the secured DPDU.

The security procedure for DPDUs that are being constructed for transmission consists of the following steps:

a) The procedure shall obtain the KeyDescriptor from Table 93 meeting the following selection criteria:

   1) The entries with KeyUsage = '0x00' (i.e., D-key). In the initial case, where a joining DLE does not have any KeyDescriptor, the joining device creates a KeyDescriptor with K_global. The KeyDescriptor shall include at least the following parameters:

      Crypto Key Identifier = 0;

      Security Level = 0x01 (MIC-32);

      KeyUsage = 0x00 (group key for PDU processing);

      Key lifetime = never-expires (0xFFFF FFFF).

   2) Of those entries, the entries valid for the current period, satisfying the inequality

      ValidNotBefore < currentTime < ValidNotAfter

      shall be selected. If none are available, the procedure shall return with a status of UNAVAILABLE_KEY.

   3) Of those entries, if two or more keys are valid for the current time, and the procedure was called from DAckPrep.Request or DAckCheck.Request, the procedure shall select the key used to authenticate the Data DPDU of the D-transaction.

      Otherwise, if two or more keys are valid for the current time, the procedure shall select the key with the larger ValidNotAfter value.

   4) Of those entries, if two or more keys have the same ValidNotAfter, the procedure shall select the key with the larger ValidNotBefore.

   5) Of those entries, if two or more keys have the same SoftExpirationTime, the procedure shall select the key with the highest Crypto Key Identifier.

b) The procedure shall retrieve the policy from the selected KeyDescriptor.

c) The procedure shall determine whether the DPDU to be secured satisfies the constraint on the maximum size of DPDUs, as follows:

   1) The procedure shall set the size M, in octets, of the DMIC authentication field from the security level.

   2) The Crypto Key Identifier Mode field in the DMXHR shall have the value 1. If the DMXHR includes the slow-channel-hopping timeslot offset field, the size of DMXHR is 3 octets; otherwise it is 2 octets.

   3) The procedure shall determine the resulting data expansion as (DMXHR_size + M).

   4) The procedure shall check whether the size of the DPDU to be secured, including data expansion, is less than or equal to the maximum DPDU size. If this check fails, the procedure shall return a status of DPDU_TOO_LONG.

d) The procedure shall use the scheduled TAI time of the start of the timeslot, as described in Table 46. If there is a potential for the device to send multiple DPDUs with the same TAI time value, then the procedure shall select a value to be conveyed in the DPDU's MHR header that is different from all other such values originated by the device at this particular value of TAI time. A procedure for determining the sequence number from which the MHR is derived is defined in 9.3.3.2.

e) The procedure shall insert the DMXHR into the DPDU outlined in Table 112, with fields set as follows:

   1) The security level subfield of the security control field shall be set to the security level 001 by default.

   2) The Crypto Key Identifier Mode subfield of the security control field shall be set to the Crypto Key Identifier Mode parameter 01 by default.

f) The procedure shall set the Crypto Key Identifier octet in the DMXHR. See Table 112.

g) The procedure shall insert the MHR sequence number in the Data DPDU MHR. See Table 110.

h) The procedure shall use the EUI64Address of the transmitting device, the 32 least significant bits of TAI time in 2^{-10} s, the low-order 3 bits of the MHR sequence number, and the channel number to build the nonce as outlined in Table 45.

i) The procedure shall use the nonce, the key material, the header, the payload and the CCM* mode of operation as described in IEEE 802.15.4:2011, 7.3.4, to secure the DPDU:

   1) If the SecurityLevel parameter specifies the use of encryption (see IEEE 802.15.4:2011, Table 58), the encryption operation shall be applied only to the DPDU's payload field. The corresponding payload field is passed to the CCM* transformation process described in IEEE 802.15.4:2011, 7.3.4, as the unsecured payload. The resulting encrypted payload shall be substituted for the original payload.

   2) The remaining fields in the DPDU, up to but not including the payload field, shall be passed to the CCM* transformation process described in IEEE 802.15.4:2011, 7.3.4, as the non-payload field.

   3) The ordering and exact manner of performing the encryption and integrity operations and the placement of the resulting encrypted data or integrity code within the DPDU payload field shall be as defined in IEEE 802.15.4:2011, 7.3.4.

j) The procedure shall return the secured DPDU and a status of SUCCESS.

7.3.2.6 Processing of received DPDUs

The inputs to the security procedure for received DPDUs are the DPDU to be unsecured, the channel number on which the DPDU was received, and the nominal TAI time of the start of the time slot in which the DPDU was received. The outputs from this procedure are the unsecured DPDU, the security level, the Crypto Key Identifier Mode, the Crypto Key Identifier, and the status of the procedure. All outputs of this procedure are assumed to be invalid unless and until explicitly set in this procedure. It is assumed that the KeyDescriptors with a single, unique device or a number of devices will have been established by the DSMO.

The security procedure on DPDU reception consists of the following steps:

a) The procedure shall set the security level and the Crypto Key Identifier Mode to the corresponding subfields of the security control field of the DMXHR of the incoming DPDU, and the Crypto Key Identifier to the corresponding subfields of the Crypto Key Identifier field of the DMXHR of the DPDU to be unsecured.

b) The procedure shall obtain the KeyDescriptor from Table 93 meeting the following selection criteria:

   1) The entries with KeyUsage = '0x00' (D-key). In the initial case, where a joining device does not have any KeyDescriptors, the joining device creates a temporary KeyDescriptor with K_global. The KeyDescriptor shall include at least the following parameters:

      CryptoKeyIdentifier = 0;

      Security Level = 0x01 (MIC-32);

      KeyUsage = 0x00 (group key for PDU processing);

      Key lifetime = never-expires (0xFFFF FFFF).

   NOTE The usage of the KeyDescriptor for K_global is described in 9.1.10.

2) Of those entries, the entry with the CryptoKeyIdentifier matching the Crypto Key Identifier of the incoming PDU shall be selected.

3) If that procedure fails, the procedure shall return with a status of UNAVAILABLE_KEY.

c) The procedure shall determine whether the security level of the incoming DPDU conforms to the security level policy by comparing the SecurityLevel of the matching KeyDescriptor obtained from step b) above. If there is a mismatch, the procedure shall return with a status of IMPROPER_SECURITY_LEVEL.

d) If the lifetime in the KeyDescriptor is finite (> 0x0000), the procedure shall verify that the 8-bit MHR sequence number has not been received previously for the same value of the source EUI64Address, the same 32-bit fixed-point fractional representation of TAI time, and the same key. If this check fails, the procedure shall return with a status of DUPLICATE_DPDU.

e) The procedure shall then use the EUI64Address of the sender, the scheduled TAI time, the low-order 3 bits of the MHR sequence number, and the channel number to generate the nonce as outlined in Table 45. Additionally, the procedure shall verify that the 8-bit MHR sequence number is not 0xFF. If the 8-bit MHR sequence number is 0xFF, the procedure shall return with a status of INVALID_SEQUENCE_NUMBER.

f) The procedure shall use the nonce, the crypto key from the KeyDescriptor obtained in step b), the actual headers (the non-payload fields), the payload and the MIC of the incoming DPDU and the CCM* mode of operation as described in the operations (see IEEE 802.15.4:2011, 7.3.5) to authenticate and, when specified, decrypt the DPDU:

1) If the security level specifies the use of encryption (see IEEE 802.15.4:2011, Table 58), the decryption operation shall be applied only to the actual DPDU payload field (see IEEE 802.15.4:2011, 5.2.2.2.2). The corresponding payload field shall be passed to the CCM* inverse transformation process described in IEEE 802.15.4:2011, 7.3.5, as the secure payload.

2) The remaining fields in the DPDU shall be passed to the CCM* inverse transformation process described in IEEE 802.15.4:2011, 7.3.5, as the non-payload fields (see IEEE 802.15.4:2011, Table 57).

3) The ordering and exact manner of performing the decryption and integrity checking operations and the placement of the resulting decrypted data within the DPDU payload field shall be as defined in IEEE 802.15.4:2011, 7.3.5.

g) If the CCM* inverse transformation process fails, the procedure shall set the unsecured DPDU to be the DPDU to be unsecured and return a status of SECURITY_ERROR.

h) If the lifetime in the KeyDescriptor is one that expires, the procedure shall insert the nonce value (includes MHR sequence number, channel number, source EUI64Address, and scheduled TAI time) in the NonceCache field of the corresponding KeyDescriptor, to enable replay protection.

i) The procedure shall return with the unsecured DPDU, the security level, the Crypto Key Identifier Mode, the Crypto Key Identifier, and a status of SUCCESS.

7.3.2.7 Detection and discard of duplicated or replayed protocol data units

See 7.3.2.6, d).

7.3.3 TL security functionality

7.3.3.1 General

The interaction of the DSC and the TL is outlined. The TL processing steps were written to reuse the commonalities between the DL and the TL. However, since the DL and the TL exist at different network abstraction layers with different requirements and assumptions, there are significant differences between the DL and TL processing steps.

Security services at the TL are selected by policy associated with the relevant transport data key, obtained as part of a new session request or a key update and based on transport policy maintained by the security manager associated with any or all of:

- the sending device;
- the requesting UAP; and/or
- the transport association, as defined by its endpoints.

The following transport security service shall always be provided with an active key:

- Authorized communication with TPDU authentication, integrity, and conveyance of the nominal time of TPDU creation, providing rejection of outdated TPDUs:
  - that were not sourced by a device within the network that shares an appropriate data key; or
  - that were severely outdated, i.e., not received at a time within DSMO.pduMaxAge seconds of the TPDU's nominal time of creation.
- Confidentiality of the application-layer payload within the TPDU.

NOTE 1 Sixteen bits of time information are transmitted with each TPDU.

NOTE 2 This service uses the originator's nominal time of transmission as authenticated at the receiver to cause rejection of TPDUs that are delayed excessively and to provide detection of duplicated TPDUs within that time window.

The confidentiality service shall not be employed with keys that are not shared secrets, because this would render true confidentiality impossible, and because this aspect of the policy associated with such keys is constant.

The MIC should be validated within the DSMO.pduMaxAge period. If the check fails, the MIC validation can be repeated by decrementing a time window to recover the creation time of the PDU.

7.3.3.2 TPDU structure

7.3.3.2.1 General

The structure of a TPDU is described in 11.5 and outlined in Figure 109 and in Figure 40 in this standard, with the TSDU possibly encrypted and the contents of the UDP header, security header, and TSDU protected by the TMIC.

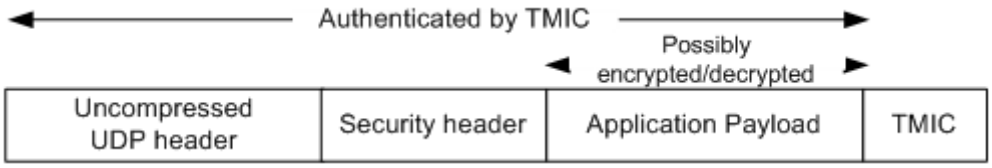


Figure 40 – TPDU structure and protected coverage

The complete TPDU from the start of the UDP header to the end of the Application Payload shall be protected by the TMIC. Each parameter in the UDP header is protected by using the transport security component (TSC) pseudo-header for the TMIC calculation. The TSC pseudo-header is described in 7.3.3.2.2.

NOTE TSC is described in 11.2. See also the brief discussion of the use of pseudo-headers in 4.5.2.1.

7.3.3.2.2 TPDU protection

NOTE 1 Use of IPV6 does not imply or provide internet connectivity. It does, however, facilitate the use of common IPvv6 management tools within the WISN.

The TMIC is used to protect the information in the UDP header, the TL security header and the TSDU. It also protects the NL source and destination IPv6 addresses by using an extended form of the UDP pseudo-header for IPv6. The UDP pseudo-header for IPv6 is

described in 11.4.2 and IETF RFC 2460:1998, 8.1. The UDP payload size and the (virtual) checksum in the UDP header are not used for the TMIC calculation.

NOTE 2 Checksum and UDP payload size do not appear in the pseudo-header since the checksum is elided (i.e., not present in the TPDU) when the TMIC is present, and the UDP payload size is determined from the NSDU size in the UDP pseudo-header for IPv6.

The parameters for the TMIC are shown in Figure 41.

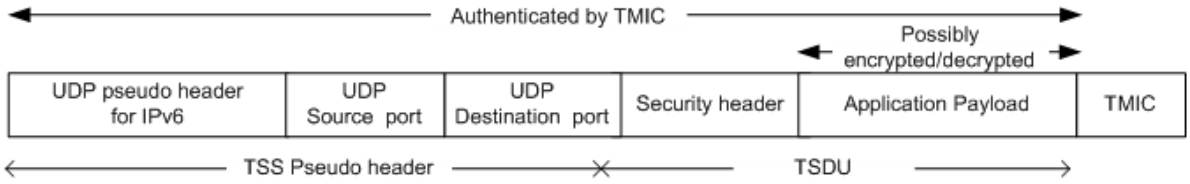


Figure 41 – TMIC parameters

The TSC constructs the TMIC parameters as outlined in Figure 41, with the received TPDU, the nominal TAI time at which the TPDU was created, the KeyDescriptor and the contract information provided by the TL. The TSC can then use the parameters for the appropriate security operation on the TPDU.

The IPv6 header and the UDP source and destination ports are passed from the TL to the TSC. The combination of those parameters is an extended UDP pseudo-header that is called the TSC pseudo-header in this standard. The structure of the TSC pseudo-header is shown in Table 47. The appropriate usage is described in 7.3.3.5.4, 7.3.3.5.5, and 7.3.3.5.8.

Table 47 – TSC pseudo-header structure

Element name	Element identifier	Element scalar type
Source IPv6Address	1	Type: IPv6Address Description: Uncompressed IPv6 address of the TPDU initiator
Destination IPv6Address	2	Type: IPv6Address Description: Uncompressed IPv6Address of the intended TPDU recipient
NSDU size	3	Type: Unsigned16 Description: NDSU size in octet
Reserved	4	Type: Unsigned8 Description: Reserved field. Currently filled with 0
Next header	5	Type: Unsigned8 Valid range: 17 (UDP) Description: Next header value in the IPv6 header. This value should be only 17
UDP source port	6	Type: Unsigned16 Description: The source UDP port number of the TPDU initiator
UDP destination port	7	Type: Unsigned16 Description: The destination UDP port number of the intended TPDU recipient

7.3.3.3 Interface with the TL for a TPDU being formed for transmission

The TL interaction with the security layer for a TPDU being formed for transmission is summarized in Figure 42. When the TSC receives the source address, source port,

destination address, destination port, and payload size, it performs a lookup in the KeyDescriptor table to see whether security is enabled for that particular session.

If the session's security level is 0: none, a header size of 3 (octets) and a TMIC size of 0 octets shall be returned.

NOTE When the security level is zero, the standard, trivially-forged UDP checksum is used to detect errors that occur during TPDU conveyance.

Otherwise the TSC shall return the appropriate Crypto Key Identifier and TMIC sizes. All sessions at the TL are unicast; therefore, the Crypto Key Identifier size shall be either 0 or 1 depending on the number of valid keys available at that time for that security association.

The TL will then call the TSC with the header and the payload. Depending on the security policy for that particular session, the payload may be encrypted, and a TMIC may be generated. The resulting header and payload will be returned to the TL for transmission.

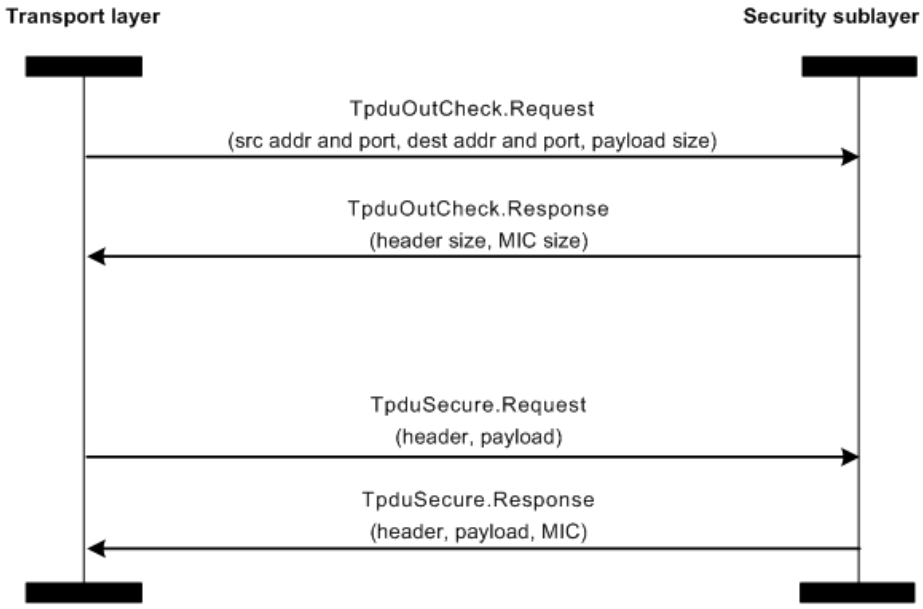


Figure 42 – TL and TSC interaction, outgoing TPDU

7.3.3.4 Processing overview for received TPDUs

The TL interaction with the security layer for a received TPDU is summarized in Figure 43. When the TSC receives the source address, source port, destination address, and destination port, it performs a lookup in the MIB to see whether security is enabled for that particular session and returns SEC_CHECK_REQUIRED or SEC_CHECK_NOT_REQUIRED.

The TL shall then call the TSC with the header, the payload, and the MIC. Depending on the security policy for that particular session, the payload may be decrypted, and a MIC may be verified. If the security check fails, a status of FAILURE will be returned, with the payload returned untouched. If the operation succeeds, the resulting payload and the recovered time of TPDU encoding will be returned to the TL.

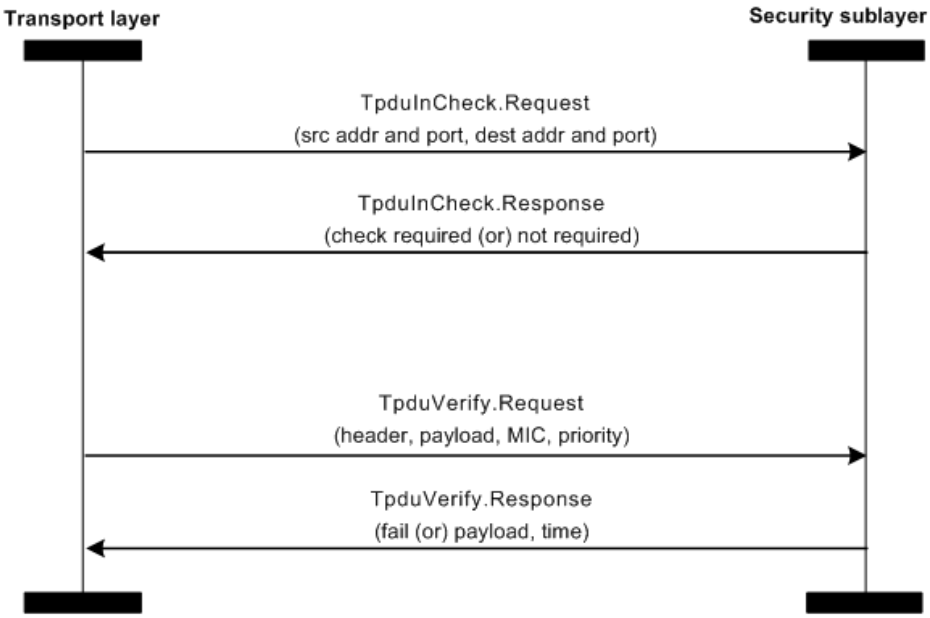


Figure 43 – TL and TSC interaction, incoming TPDU

7.3.3.5 TL interface to the TSC

7.3.3.5.1 General

The relationship between the TL and the TSC is outlined in 7.3.3.2 for the TL interface for an outgoing TPDU and 7.3.3.4 for the TL interface for an incoming TPDU.

7.3.3.5.2 Sec.TpduOutCheck.Request

7.3.3.5.2.1 General

Sec.TpduOutCheck.Request is a check from the TL to the TSC to obtain the size of the security fields (if any) required in the outgoing TPDU.

7.3.3.5.2.2 Semantics of the service primitive

The semantics of Sec.TpduOutCheck.Request are as follows:

```
Sec.TpduOutCheck.Request (
    Source_Address,
    Source_Port,
    Destination_Address,
    Destination_Port,
    Payload_Size)
```

Table 48 specifies the elements for the Sec.TpduOutCheck.Request.

Table 48 – Sec.TpduOutCheck.Request elements

Element name	Element identifier	Element scalar type
Source_Address	1	Type: IPv6Address Valid range: all with high-order bit reset
Source_Port	2	Type: Integer
Destination_Address	3	Type: IPv6Address
Destination_Port	4	Type: Integer
Payload_Size	5	Type: Integer Valid range: 0..Assigned_Max_TSDU_Size; see 11.4.3.3

The TSC shall use the Source_Address, Source_Port, Destination_Address, and Destination_Port to retrieve the appropriate policy (if any) for this security association.

7.3.3.5.2.3 Appropriate usage

The TL invokes the Sec.TpduOutCheck.Request primitive to protect a TPDU before it is transmitted.

7.3.3.5.2.4 Effect on receipt

On receipt of the Sec.TpduOutCheck.Request primitive, the TSC determines if the TPDU needs to be protected and returns the corresponding header and TMIC sizes.

7.3.3.5.3 Sec.TpduOutCheck.Response

7.3.3.5.3.1 General

Sec.TpduOutCheck.Response reports the result of a Sec.TpduOutCheck.Request.

7.3.3.5.3.2 Semantics

The semantics of Sec.TpduOutCheck.Response are as follows:

```
Sec. TpduOutCheck.Response      (
    Sec_Header_Size,
    TMIC_Size)
```

Table 49 specifies the elements for Sec.TpduOutCheck.Response.

Table 49 – Sec.TpduOutCheck.Response elements

Element name	Element identifier	Element scalar type
Sec_Header_Size (the additional header size required by the TSC, in full octets)	1	Type: Integer Valid range: 0..Assigned_Max_TSDU_Size; see Table 30
TMIC_Size (the size of the Transport Integrity Code, in full octets)	2	Type: Integer Valid values: 0, 4, 8 or 16

7.3.3.5.3.3 When generated

The TSC generates Sec.TpduOutCheck.Response in response to a Sec.TpduOutCheck.Request. The Sec.TpduOutCheck.Response returns the additional sizes

required to support the security layer functionality. A security association is a secure TL association based on:

- the source address;
- the source port;
- the destination address;
- the destination port.

7.3.3.5.3.4 Appropriate usage

On receipt of Sec.TpduOutCheck.Response, the TL is notified of the need to apply a security operation on the TPDU, along with the additional octets required to support that operation.

7.3.3.5.4 Sec.TpduSecure.Request

7.3.3.5.4.1 General

Sec.TpduSecure.Request instructs the TSC to carry out the appropriate steps to secure an outgoing TPDU. The information about the security association is contained in the pseudo-header passed to the TSC. See Figure 108 in 11.4.2.

7.3.3.5.4.2 Semantics of the service primitive

The semantics of Sec.TpduSecure.Request are as follows:

```
Sec. TpduSecure.Request    (
    TSC_Pseudo_Header,
    TSC_Pseudo_Header_Size,
    TSDU,
    TSDU_Size)
```

Table 50 specifies the elements for the Sec.TpduSecure.Request.

Table 50 – Sec.TpduSecure.Request elements

Element name	Element identifier	Element scalar type
TSC_Pseudo_Header	1	Type: OctetStringN
TSC_Pseudo_Header_Size	2	Type: Integer Valid range: 0..127
TSDU	3	Type: OctetStringN
TSDU_Size	4	Type: Integer Valid range: 0..Assigned_Max_TSDU_Size; see Table 30

The TSC shall obtain the source address, source port, destination address, and destination port from the TSC pseudo-header. That information is used to retrieve the appropriate keying material and policies for this security association.

The TSC includes the Priority information in the TPDU header, provides data confidentiality for the TPDU, and generates the TMIC field as dictated by the PDU processing steps and current policies.

The TSC populates the TL security header as specified in the TPDU processing steps and policies.

7.3.3.5.4.3 Appropriate usage

The TL invokes the Sec.TpduSecure.Request primitive to protect an outgoing TPDU after the TL receives the number of additional octets required in the transport header and the TMIC.

7.3.3.5.4.4 Effect on receipt

On receipt of the Sec.TpduSecure.Request primitive, the TSC starts the appropriate PDU processing steps to protect the outgoing TPDU as dictated by the outgoing TPDU processing steps in 7.3.3.8.

7.3.3.5.5 Sec.TpduSecure.Response

7.3.3.5.5.1 General

Sec.TpduSecure.Response reports the result of a Sec.TpduSecure.Request.

7.3.3.5.5.2 Semantics

The semantics of Sec.TpduSecure.Response are as follows:

```
Sec. TpduSecure.Response (
    TSC_Pseudo_Header,
    TSC_Pseudo_Header_Size,
    TSDU,
    TSDU_Size,
    TMIC,
    TMIC_Size,
    Status)
```

Table 51 specifies the elements for Sec.TpduSecure.Response.

Table 51 – Sec. TpduSecure.Response elements

Element name	Element identifier	Element scalar type
TSC_Pseudo_Header	1	Type: OctetStringN
TSC_Pseudo_Header_Size	2	Type: Unsigned7 Valid range: 0..127
TSDU	3	Type: OctetStringN
TSDU_Size	4	Type: Unsigned Valid range: 0..Assigned_Max_TSDU_Size; see Table 30
TMIC	5	Type: OctetStringN
TMIC_Size	6	Type: Integer Valid values: 0, 4, 8, 16
Status	7	Type: Unsigned Named values: 0: success; > 0: failure

7.3.3.5.5.3 When generated

The TSC generates Sec.TpduSecure.Response in response to a Sec.TpduSecure.Request. The Sec.TpduSecure.Response returns a populated security transport header, a possibly encrypted TSDU and a TMIC along with the appropriate sizes. Finally the

Sec.TpduSecure.Response returns a status value that indicates either SUCCESS or the appropriate error code.

7.3.3.5.5.4 Appropriate usage

On receipt of Sec.TpduSecure.Response, the TL is notified of the result of protecting an outgoing TPDU.

7.3.3.5.6 Sec.TpduInCheck.Request

7.3.3.5.6.1 General

Sec.TpduInCheck.Request instructs the TSC to verify and possibly decrypt an incoming TL protocol data unit as appropriate.

7.3.3.5.6.2 Semantics of the service primitive

The semantics of Sec. TpduInCheck.Request are as follows:

```
Sec.TpduInCheck.Request   (
    Source_Address,
    Source_Port,
    Destination_Address,
    Destination_Port,
    Payload_Size)
```

Table 52 specifies the elements for the Sec.TpduInCheck.Request.

Table 52 – Sec.TpduInCheck.Request elements

Element name	Element identifier	Element scalar type
Source_Address	1	Type: IPv6Address
Source_Port	2	Type: Unsigned16
Destination_Address	3	Type: IPv6Address
Destination_Port	4	Type: Unsigned16
Payload_Size	5	Type: Unsigned Valid range: 0..Assigned_Max_TSDU_Size

The TSC uses the Source_Address, Source_Port, Destination_Address, and Destination_Port to retrieve the appropriate policy (if any) for this security association.

7.3.3.5.6.3 Appropriate usage

The TL invokes the Sec.TpduInCheck.Request primitive to check if a secure verification and possibly a decryption of a TPDU beforehand are required.

7.3.3.5.6.4 Effect on receipt

On receipt of the Sec.TpduInCheck.Request primitive, the TSC determines if the TPDU needs to be verified and, potentially, decrypted and returns a status of success or failure.

7.3.3.5.7 Sec. TpduInCheck.Response

7.3.3.5.7.1 General

Sec.TpduInCheck.Response reports the result of a Sec.TpduInCheck.Request.

7.3.3.5.7.2 Semantics

The semantics of Sec.TpduInCheck.Response are as follows:

```
Sec.TpduInCheck.Response (
    Status)
```

Table 53 specifies the elements for Sec.TpduInCheck.Response.

Table 53 – Sec.TpduInCheck.Response elements

Element name	Element identifier	Element scalar type
Status (the result of a Sec.TpduInCheck.Request primitive)	1	Type: Unsigned Named values: 0: success; > 0: failure

7.3.3.5.7.3 When generated

The TSC generates Sec.TpduInCheck.Response in response to a Sec.TpduInCheck.Request. The Sec.TpduInCheck.Response returns a status value that indicates either TRUE or FALSE depending on the policies on the current security association.

7.3.3.5.7.4 Appropriate usage

On receipt of Sec.TpduInCheck.Response, the TL is notified of the need to call the Sec.TpduVerify.Request to verify and possibly decrypt the incoming TPDU.

7.3.3.5.8 Sec.TpduVerify.Request

7.3.3.5.8.1 General

Sec.TpduVerify.Request instructs the TSC to verify and, where so configured, decrypt an incoming TPDU.

7.3.3.5.8.2 Semantics of the service primitive

The semantics of Sec.TpduVerify.Request are as follows:

```
Sec.TpduVerify.Request    (
    TSC_Pseudo_Header,
    TSC_Pseudo_Header_Size,
    TSDU,
    TSDU_Size,
    TMIC,
    TMIC_Size)
```

Table 54 specifies the elements for the Sec.TpduVerify.Request.

Table 54 – Sec.TpduVerify.Request elements

Element name	Element identifier	Element scalar type
TSC_Pseudo_Header	1	Type: OctetString
TSC_Pseudo_Header_Size	2	Type: Unsigned Valid range: 0..127
TSDU	3	Type: OctetString
TSDU_Size	4	Type: Unsigned Valid range: 0..Assigned_Max_TSDU_Size; see Table 30
TMIC	5	Type: OctetString
TMIC_Size	6	Type: Integer Valid values: 0, 4, 8 or 16
Priority	7	Type: Unsigned4

The TSC verifies that the TL Security Header of the TPDU has the appropriate security control (octet 1) by comparing it to the current policy. The Crypto Key Identifier (octet 2), if present, is used to retrieve the correct key material. See 7.3.3.9.

The TSC verifies the TMIC field as dictated by the current policies.

The time conveyed in the TL security header is used in the nonce construction for the authentication and, where configured, decryption of the received TPDU.

The priority is provided to the TSC in order to allow efficient implementation of replay protection.

7.3.3.5.8.3 Appropriate usage

The TL invokes the Sec.TpduVerify.Request primitive to verify and possibly decrypt a TPDU before it is transmitted.

7.3.3.5.8.4 Effect on receipt

On receipt of the Sec.TpduVerify.Request primitive, the TSC starts the appropriate TPDU processing steps to verify and, where configured, decrypt the received TPDU as dictated by the processing steps for received TPDUs in 7.3.3.9.

7.3.3.5.9 Sec.TpduVerify.Response

7.3.3.5.9.1 General

Sec.TpduVerify.Response reports the result of a Sec.TpduVerify.Request.

7.3.3.5.9.2 Semantics

The semantics of Sec.TpduVerify.Response are as follows:

```
Sec.TpduVerify.Response    (
    TSDU,
    TSDU_Size,
    Time_Of_TPDU_Creation,
    Status)
```

Table 55 specifies the elements for Sec.TpduVerify.Response.

Table 55 – Sec.TpduVerify.Response elements

Element name	Element identifier	Element scalar type
TSDU (after any required decryption)	1	Type: OctetString
TSDU_Size	2	Type: Unsigned Valid range: 0..Assigned_Max_TSDU_Size; see 11.4.3.3
Time_Of_TPDU_Creation (32-bit fixed-point fractional representation of TAI time, modulo 2^{22} s, used in the nonce)	3	Type: Unsigned32
Status	4	Type: Unsigned Named values: 0: success; > 0: failure

7.3.3.5.9.3 When generated

The TSC generates Sec.TpduVerify.Response in response to a Sec.TpduVerify.Request. The Sec.TpduVerify.Response returns a status value that indicates either SUCCESS or the appropriate error code.

7.3.3.5.9.4 Appropriate usage

On receipt of Sec.TpduVerify.Response, the TL is notified of the result of verifying and possibly decrypting the incoming TPDU.

7.3.3.6 TPDU security header structure

The TPDU security header structure is as described in Table 56.

Table 56 – Structure of TL security header

Octet	Bits							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	Security_Control							
2 (opt)	Crypto_Key_Identifier							
3	Nominal_Time							
4								

The TL security header shall be added to all TPDUs to use the header compression in the NL. In the case that no KeyDescriptor corresponds to a specific TPDU, that TPDU shall be treated as if it had a security level of NONE.

NOTE 1 When the TPDU has no TMIC, the UDP checksum is used for error detection.

Fields include:

- Security_Control: as defined in 7.3.1.2.
- Crypto_Key_Identifier: specifies the current Crypto Key Identifier used to protect this TPDU.
- Nominal_Time: the time portion shall be 16 bits of TAI time, expressed modulo 2^6 s in units of 2^{-10} s, presented in MSB order.

NOTE 2 Fixing the time granularity to 2^{-10} s gives the TL the ability to transmit 1 023 TPDUs per second. With the maximum payload size of a TPDU, this is adequate for the throughput specified by 6LoWPAN. A varying granularity for TPDU time, suitable for supporting higher-rate automation processes, is a possible area of future standardization.

7.3.3.7 Nonce construction for TPDUs

This standard uses a different (but related) TPDU nonce construction than that of its DPDUs. A 13-octet nonce is required for the CCM* engine. The nonce shall be constructed as the concatenation from first (leftmost) to last (rightmost) octets of data fields as shown in Table 57, wherein:

- the EUI64Address shall be used as an array of 8 octets and the truncated TAI time;
- the nominal TAI time of the TPDU creation shall be set at a granularity of 2^{-10} s, and shall be no more than 1 s earlier than the actual local time of start of TPDU creation, and no more than 1 s later than the actual local time of end of TPDU creation. Each outgoing TPDU from a given source EUI64Address that uses a given key shall be created with a unique value for the 32-bit truncated nominal TAI time of TPDU creation. This encoding restricts the maximum data rate of the TL to 1 024 TPDUs per second, which shall not be exceeded. The structure of the 32-bit truncated nominal TAI time shall be as described in Table 58.

Table 57 – Structure of the TPDU nonce

Octet	Bits							
	7	6	5	4	3	2	1	0
1	EUI64Address							
…								
8								
9	Truncated nominal TAI time of TPDU creation							
…								
12								
13	0xFF							

The 32-bit truncated nominal representation of TAI time used in the T-nonce is described in Table 58.

Table 58 – Structure of 32-bit truncated nominal TAI time used in the T-nonce

Octet	Bits							
	7	6	5	4	3	2	1	0
1	Nominal TAI time (bits with weight $2^{21}..2^{14}$ s)							
2	Nominal TAI time (bits with weight $2^{13}..2^{6}$ s)							
3	Nominal TAI time (bits with weight $2^{5}..2^{-2}$ s)							
4	Nominal TAI time (bits with weight $2^{-3}..2^{-10}$ s)							

At a 2^{-10} s granularity, there will be 48,5 days before this 32-bit time representation repeats, thus providing at most 48,5 days before a new key needs to be deployed to avoid a potential nonce collision with resultant keystream reuse.

NOTE This representation is chosen because the sender and intended receivers are presumed to share approximately the same time sense and same nominal start time for any MAC transaction timeslot.

7.3.3.8 Processing for TPDUs to be transmitted

The inputs to the security procedure for TPDUs to be transmitted are:

- the TPDU to be secured;
- the EUI64Address of the source device;
- the nominal TAI time;
- the source and destination IPv6Addresses; and
- the source and destination port.

The outputs from this procedure are:

- the status of the procedure; and
- if this status is SUCCESS, the secured TPDU.

The security procedure for TPDUs being constructed for transmission consists of the following steps:

a) The procedure shall obtain the KeyDescriptor from Table 93 meeting the following selection criteria:

   1) The entries with Type = 10 (TL). If none are available, the procedure shall return with a status of UNAVAILABLE_KEY.

   2) Of those entries, the entries with the KeyLookupData matching the SourceAddress||SourcePort||DestinationAddress||DestinationPort (see Table 94) for this TPDU. If no KeyDescriptor is available and the following two conditions are both true, the procedure shall treat the TPDU as a no-security (security level = NONE) TPDU. Otherwise, the procedure shall return with a status of UNAVAILABLE_KEY.

   3) Condition1: The join state of the receiving device is Provisioned or Joining (see Table 79).

   4) Condition2: Source and destination ports are both for the DMAP (i.e., 0xF0B0).

      Those conditions need to be satisfied to transmit a join TPDUs that has a security level of NONE.

   5) Of those entries, the entries valid for the current period, satisfying the inequality ValidNotBefore < current time < ValidNotAfter shall be selected. If none are available, the procedure shall return with a status of UNAVAILABLE_KEY.

   6) Of those entries, if two or more keys are valid for the current time, the procedure shall select the key with the longest ValidNotAfter value.

   7) Of those entries, if two or more keys have the same ValidNotAfter, the procedure shall select the key with the smallest ValidNotBefore.

   8) Of those entries, if two or more keys have the same ValidNotBefore, the procedure shall select the key with the highest Crypto Key Identifier.

      If the procedure fails, the procedure shall handle the TPDU as no security (security level = NONE).

b) The procedure shall retrieve the policy from the selected KeyDescriptor.

c) The procedure shall determine whether the TPDU to be secured satisfies the constraint on the maximum size of TPDUs, as follows:

   1) The procedure shall set the size M, in octets, of the TMIC authentication field from the security level.

   2) The size of the Key Index field in the TL security header shall be 1 octet, if more than 1 key is valid for the current security association and 0 otherwise.

   3) The procedure shall determine the data expansion as Crypto Key Identifier size + M.

4) The procedure shall check whether the size of the TPDU to be secured, including data expansion, is less than or equal to the Assigned_Max_TSDU_Size (see Table 30). If this check fails, the procedure shall return a status of TPDU_TOO_LONG.

d) The procedure shall build the security control octet of the TL security header. If the security level matches more than one KeyDescriptor from the current Key Descriptor, the Crypto Key Identifier shall be used with the Crypto Key Identifier Mode = 0x01; otherwise the procedure shall set the Crypto Key Identifier Mode = 0x00.

e) The procedure shall set the Crypto Key Identifier = Crypto Key Identifier from the current Key Descriptor (if present) in the TL security header. See Table 56.

f) The procedure shall build the nominal TAI time in TAITimeRounded format as outlined in Table 58.

g) The procedure shall set the Nominal_Time octets in the outgoing TL security header as the last 16 bits of the nominal TAI time in TAITimeRounded format. See octets 3 and 4 in Table 58.

h) If no Key Descriptor was found, then go to step j); otherwise, the procedure shall use the EUI64Address, the nominal TAI time in TAITimeRounded format and the 8-bit value 0xFF to build the nonce as outlined in Table 57.

i) The procedure shall use the nonce, the key material, the TPDU header, the TPDU payload and the CCM* mode of operation as described in IEEE 802.15.4:2011, 7.3.4, to secure the TPDU:

1) If the SecurityLevel parameter specifies the use of encryption (see IEEE 802.15.4:2011, Table 58), the encryption operation shall be applied only to the TPDU's payload field. The corresponding payload field is passed to the CCM* transformation process described in IEEE 802.15.4:2011, 7.3.4, as the unsecured payload. The resulting encrypted payload shall be substituted for the original payload.

2) The remaining fields in the TPDU, up to but not including the payload field, plus any required virtual fields, shall be passed to the CCM* transformation process described in IEEE 802.15.4:2011, 7.3.4, as the non-payload field.

3) The ordering and exact manner of performing the encryption and integrity operations and the placement of the resulting encrypted data or integrity code within the TPDU payload field shall be as defined in IEEE 802.15.4:2011, 7.3.4.

j) The procedure shall return the secured TPDU and a status of SUCCESS.

7.3.3.9 Processing for received TPDUs

The input to the security procedure for received TPDUs is the TPDU to be unsecured, which contains the source and destination IPv6Addresses and the source and destination ports. The outputs from this procedure are the unsecured TPDU, the security level, the Crypto Key Identifier Mode, the key source, the key index, and the status of the procedure. All outputs of this procedure are assumed to be invalid unless and until explicitly set in this procedure. Each receiver of TPDUs maintains a cache of authenticated nonce values of recently received TPDUs.

The security procedure on TPDU reception consists of the following steps:

a) The procedure shall obtain the security level and the Crypto Key Identifier Mode from the corresponding subfields of the security control field and the key index from the corresponding subfields of the Crypto Key Identifier (if present) of the security header of the incoming TPDU.

b) The procedure shall reconstruct the inferred originator's nominal TAI time of TPDU formation (see Note 2).

c) The procedure shall compare the time in step b) and the receiver's current TAI time. If the time in step b) is more than 2 s ahead of the receiver's current TAI time, or more than N seconds behind the receiver's current TAI time (where N is a policy-determined parameter whose default value is 62 s) the security process returns FAILURE_TPDU_DID_NOT_AUTHENTICATE.

d) The procedure shall obtain the KeyDescriptor from Table 93 meeting the following selection criteria:

1) The entries with Type = 10 (TL).

2) Of those entries, the entries with the KeyLookupData matching the SourceAddress||SourcePort||DestinationAddress||DestinationPort (see Table 94) for this TPDU. If no KeyDescriptor is available and the following two conditions are all true, the procedure shall treat the TPDU as a no security (security level = NONE) TPDU. Otherwise, the procedure shall return with a status of UNAVAILABLE_KEY.

i) Condition1: The join state of the receiving device is Provisioned or Joining (see Table 79).

ii) Condition2: Source and destination ports are both for DMAPs (i.e. 0xF0B0).

NOTE 1 This information is used when processing a received join TPDU that has a security level of NONE.

3) Of those entries, the entries valid for the current period, satisfying the inequality ValidNotBefore < current time < ValidNotAfter shall be selected. If none is available, the procedure shall return with a status of UNAVAILABLE_KEY.

4) Of those entries, if two or more keys are valid for the current time, the procedure shall select the key with the longest ValidNotAfter value.

5) Of those entries, if two or more keys have the same ValidNotAfter, the procedure shall select the key with the smallest ValidNotBefore.

6) Of those entries, if two or more keys have the same ValidNotBefore, the procedure shall select the key with the highest Crypto Key Identifier.

7) If the procedure fails, the procedure shall return with a status of UNAVAILABLE_KEY.

e) The procedure shall determine whether the security level of the incoming TPDU conforms to the security level policy by comparing the SecurityLevel of the matching Key Descriptor obtained from step b) above. If there is a mismatch, the procedure shall return with a status of IMPROPER_SECURITY_LEVEL.

f) The procedure shall then use the EUI64Address of the originator, the nominal TAI time of TPDU formation from step b), the received low-order 16 bits of nominal TAI time (see Table 58) to generate the nonce outlined in Table 57.

g) The procedure shall use the nonce, the key from the Key Descriptor obtained in step d), the headers (the non-payload fields), the payload and the MIC of the incoming TPDU and the CCM* mode of operation as described in the operations (see IEEE 802.15.4:2011, 7.3.5) to authenticate and, where configured for the transport association, decrypt the TPDU:

1) If the security level specifies the use of encryption (see IEEE 802.15.4:2011, Table 58), the decryption operation shall be applied only to the actual TPDU payload field (see IEEE 802.15.4:2011, 5.2.2.2.2). The corresponding payload field shall be passed to the CCM* inverse transformation process described in IEEE 802.15.4:2011, 7.3.5 as the secure payload.

2) The remaining fields in the TPDU, plus any required virtual fields, shall be passed to the CCM* inverse transformation process described in IEEE 802.15.4:2011, 7.3.5 as the non-payload fields (see IEEE 802.15.4:2011, Table 57).

3) The ordering and exact manner of performing the decryption and integrity checking operations and the placement of the resulting decrypted data within the TPDU payload field shall be as defined in IEEE 802.15.4:2011, 7.3.5.

h) If the CCM* inverse transformation process fails, then the procedure may decrement the nominal TAI time by 64 s and repeat the above process during DSMO.pduMaxAge period. Otherwise, the procedure shall set the TPDU to be unsecured and return a status of SECURITY_ERROR.

i) The procedure shall look up the nonce of the just authenticated TPDU in the cache, with the following possible outcomes:

1) The time of the nonce is older than the oldest time in the cache and the cache does not have space for an additional older entry, so the security process returns FAILURE_OVERAGE_TPDU.

2) The nonce is already in the cache, so the security process returns FAILURE_DUPLICATE_TPDU.

3) Any encrypted payload of the TPDU is decrypted, and the security process returns SUCCESS.

j) The procedure shall insert the nonce value in the cache, if necessary bumping from the cache the cache entry with the oldest inferred nominal TAI time of TPDU formation and return with the unsecured TPDU, the security level, the Crypto Key Identifier Mode, the key source, the key index (if present) and a status of SUCCESS.

NOTE 2 The originator's nominal TAI time of TPDU formation is inferred initially from the receiver's current TAI time and the fractional nominal TAI time of TPDU creation specified in the TPDU such that it satisfies the relationship

((current-receiver-time + 2 s) ≥ originator's-nominal-time ≥ (current-receiver-time – DSMO.pduMaxAge)).

The time duration of 2 s is intended to cover ±1 s boundary conditions.

It is permitted for the cache to be segmented into separate caches for each sending EUI64Address. It is further permitted for the cache to be segmented by the reported network-layer QoS, so that a cache that holds only a few nonces of low-priority TPDUs need not also hold dozens to hundreds of nonces for overtaking higher-priority TPDUs. It is further permitted for the cache size to be adaptive, so that repeated occurrences of the first outcome in step g) above cause the cache to grow, with appropriate reduction in cache size if and when the excess cache capacity has not been used for an extended period of time.

7.3.3.10 Detection and discard of duplicated or replayed TPDUs

See 7.3.3.9, i).

7.4 Joining process

7.4.1 General

The joining process describes the steps by which a new device is admitted into a standard-compliant network and obtains all the relevant information to be able to communicate with other devices as well as the system manager and security manager.

NOTE This description assumes that the joining device has a DL-protocol stack conforming to some edition of this standard. However, since this procedure is an AL protocol, it is also usable for devices that do not have a DL stack simply by omitting the DL steps.

7.4.2 Prerequisites

The joining process follows the provisioning step, during which cryptographic information and non-cryptographic configuration parameters may be provided to the new device. A new device shall obtain such necessary provisioning information from the provisioning device. This is described in Clause 13. The Join_Command attribute in the DMO of a device shall be used to command the device to join the network.

A joining device shall join the target network using one of the following security approaches:

- symmetric keys;
- asymmetric keys;
- no-security.

The no-security approach does not use a secret key for transfer of join keys. Instead, it uses one of the predefined, well-known keys K_global or K_open, as specified in 7.2.2.2. In this case the MIC functions as a strong CRC, which offers no security assurances but has a very

high probability of detection of errors not due to deliberate attack. In this case end-to-end secure sessions (T-associations) are not permitted.

A device implementing the symmetric-key join approach shall have both a symmetric join key and the EUI64Address of a security manager that shares that join key.

A device implementing the asymmetric-key join approach shall have a certificate signed by a certificate authority trusted by the target network.

A device implementing the no-security join approach shall have the well-known, published, non-secret symmetric key common to all standard-compliant networks, K_global or K_open, as specified in 7.2.2.2.

7.4.3 Desired device end state and properties

At the conclusion of the joining process, the system shall have the following state:

- the new device and the security manager securely share a symmetric long-term master key;
- if a WISN DLE is present in the device, the new device has the required cryptographic material for that DLE to exchange DPDUs with its direct neighbors;
- if a WISN DLE is present in the device, the new device has the required non-cryptographic material and resources for that DLE to exchange DPDUs with at least one of its direct neighbors; and
- the new device shall have a contract with the system manager.

NOTE 1 A contract with the system manager includes a T-key shared between the system manager and a TLE of the new device.

When using either the symmetric-key or asymmetric-key approach, the joining process provides the following security assurances:

- protection against replay attacks on join APDUs:
  - cryptographic assurance to the new device that the security manager is alive;
  - cryptographic assurance to the security manager that the new device is alive;
- authenticity:
  - cryptographic assurance that the join request comes from a device that has valid trust material;
  - cryptographic assurance that the join APDUs have not been altered;
- confidentiality:
  - cryptographic protection for the keys in the join reply, such that an eavesdropper cannot recover the transported keys.

NOTE 2 A challenge/response protocol is used for the secure joining process to eliminate any need to rely, at the time of the joining process, on a mutually trusted source of TAI time.

7.4.4 Joining process steps common for symmetric-key and asymmetric-key approaches

7.4.4.1 General

When using the secure symmetric-key or asymmetric-key approach for the joining process, the device goes through the following general steps to complete the joining process.

The system manager controls the process of a new device joining the network. A non-joined device that implements a DLE conforming to this standard listens for advertisement DPDUs from local routers, whose advertisement functions are configured by the system manager.

Advertisements can be found by using active scanning, passive scanning, or a combination of both. Active scanning involves solicitation DPDUs sent by the joining device to request the transmission of advertisement DPDUs. Detailed information for active/passive scanning is found in 9.1.13.

Such an advertising router shall assist during the joining process by acting as a proxy for a system manager, relative to the new device. As a proxy, this advertising router forwards the join request from the new device to a system manager and forwards the join response from that system manager to the new device. Upon receipt of a join request from a new device, a system manager processes the request, authenticates the acceptability of the request through communication with a security manager, and generates a join response in reply.

7.4.4.2 Construction of joining process PDUs

The join request from a new device consists of concatenated PDUs that separate the security information, exchanged between the device and a security manager, from the non-security information, exchanged between the device and a system manager. The join response consists of similar concatenated PDUs that separate the security information from the non-security information.

The non-security information exchanged during the joining process is described in 6.3.9.2. That exchange uses the method defined in 6.3.9.2.2 for the advertising router's DMO and methods defined in 6.3.9.5 for the system manager's DMSO.

The security information exchanged during the joining process is dependent on the join approach used, as described in 7.4.5 for the symmetric-key approach and in 7.4.6 for the asymmetric-key approach.

In order for the new device to construct its joining process PDUs and send them to the advertising router, it needs to know the EUI64Address of the advertising router. The process followed by the new device to obtain this EUI64Address is described in 9.1.14. The joining process request PDUs, from the new device to the advertising router, and the joining process response PDUs, from the advertising router to the new device, shall be constructed as follows, using this EUI64Address of the advertising router and the EUI64Address of the new device:

- If there is a WISN DLE present in the joining device, the DL header for these joining process PDUs is constructed as described in 9.3.
- The NL header for these joining process PDUs is constructed as described in 10.5.3.
- The TL header for these joining process PDUs is constructed as described in 11.5. For calculation of the UDP checksum, the UDP pseudo-header for IPv6 uses the EUI64Address of the new device and the EUI64Address of the advertising router, represented as link-local IPv6Addresses of the respective devices, as the IPv6Addresses of the PDUs' source and destination.
- The DMAPs of the joining device and of the advertising router use these link-local IPv6Addresses when conveying join-request-related PDUs to their TLEs.
- At the TL, the Sec.TpduOutCheck.Response shall return with a value of 3 for the security header size and 0 for the TMIC size, indicating a TL security header with security level = 0 (NONE). Due to a (usual) lack of shared secret keys, TL security protection is not generally available for the TPDU exchange of the join request PDUs from the new device and the join response PDUs from the advertising router.
- If there is a DLE present, the DPDU security header has a security level = 1 (MIC-32), thus using a 32-bit DMIC for join DPDUs, which shall be constructed as described in 7.3.2.5 using the D-key K_global (Crypto Key Identifier = 0). Because this key is well-known, it provides no protection against deliberate attack. Thus this 32-bit DMIC is used only to detect unintentional errors in join request and response DPDUs. The advertising D-router shall use its existing contract with the system manager to forward the joining process PDUs on behalf of the device that is making the join request.

NOTE A new device that is trying to join the network does not have any contracts assigned by the system manager. Thus the communication between the new device and the advertising router is not based on any contract.

The PSMO.Security_Sym_Confirm() request and response messages should be protected by the D-key and T-key information that is distributed in the PSMO.Proxy_Security_Sym_Join() response message.

7.4.4.3 Protection of joining process messages

7.4.4.3.1 General

As the new device does not have the necessary D-subnet key and a TL level T-key with the advertising router, all joining process messages, other than confirm messages, between the new device and the advertising router shall use the K_global at the DL level to construct a 32-bit DMIC. At the TL level, the UDP checksum shall be used for these messages.

The security information in the join request, as well as all the information coming back from the system manager and the security manager in the join response messages, is protected using the join key. This is described in 7.4.5 for the symmetric-key approach and in 7.4.6 for the asymmetric-key approach.

7.4.4.3.2 Protection against join PDU replay attacks

To protect against a join PDU replay attack, it is recommended that the security manager check for duplicate challenges with a valid MIC from the new device. If no challenge duplicates are detected, the security manager stores the challenge value for further duplication checking. In the event of a duplicate detection, the security manager discards the PDU before processing the join PDU.

7.4.4.3.3 Protecting non-security message in the joining process

7.4.4.3.3.1 General

Subclause 7.4.4.3.3 describes the configuration of the security settings during the joining process. The non-security related network information is configured with the DMSO.System_Manager_Join() and DMSO.System_Manager_Contract() methods. The details of the methods are specified in 6.3.9.2. Such non-security messages are protected with cryptographic operations at the AL. The non-security messages are generated in the system manager and passed to the security manager which then adds the MIC using the join key. The protected messages are transmitted to the joining device via the DMSO in the system manager. At the joining device, the MIC in the received response is validated with the same operation.

At the joining device, the MIC in the received response is validated with same operation.

7.4.4.3.3.2 MIC generation for System_Manager_Join response

The DMSO.System_Manager_Join() method is defined in 6.3.9.5. The MIC field is the most significant 4 octets in MACTag generated with the following operation:

$$MACTag = HMAC\text{-}MMO_{K\_join}[\text{Output Argument number1 .. number6 in Table 23} \,||\, EUI64Address_{join\_device} \,||\, Challenge_{join\_device}].$$

7.4.4.3.3.3 MIC generation for System_Manager_Contract response

The DMSO.System_Manager_Contract method is defined in 6.3.9.5. The MIC field is the most significant 4 octets in MACTag generated with the following operation:

$$MACTag = HMAC\text{-}MMO_{K\_join}[\text{Output Argument number1 in Table 24} \,||\, EUI64Address_{join\_device} \,||\, Challenge_{join\_device}]$$

7.4.4.3.3.4 Confirmation

After the DMO.Proxy_System_Manager_Join().Response and DMO.Proxy_System_Manager_Contract().Response are received, the join device sends a message to inform that the correct network information has been received by the system manager.

In the symmetric-key joining process, the confirmation process is integrated in the PSMO.Security_Confirm() method specified in Table 61.

In the asymmetric-key joining process, the confirmation process is accomplished with the PSMO.Network_Information_Confirmation() method specified in Table 74.

7.4.4.4 Join timers

In the joining process, two timers are defined. Upon expiration of either of those timers, any information (e.g., state and received parameter) cached for the particular joining process shall be removed or re-initialized.

JT_1: Time duration managed in the joining device, from the time of transmission of the security join request, to the time of correct validation of the confirmation response generated by the security manager. If the joining device is not a backbone device, the initial value for JT_1 shall be set at the DauxJoinTimeout value that is distributed within the DL advertisement (see Table 127). Otherwise, the initial value for JT_1 shall be set to 60 s for the backbone device.

JT_2: Time duration managed in the security manager, from the time of reception of the security join request to the time of correct validation of the security confirmation message generated by the joining device. The actual value of JT_2 is not specified in this standard.

NOTE JT_2 can be less than JT_1.

7.4.4.5 Joining process of backbone device

A backbone device joins a target network by executing the join method in the system manager's DMO instead of the advertisement router's. Therefore, the backbone device does not need to discover an advertisement router; the DPO.Target_System_Manager_Address shall be set in the provisioning phase. The overview of the backbone device joining process is illustrated in Figure 45 for the symmetric-key joining process and in Figure 48 for the asymmetric-key joining process.

7.4.4.6 TMIC size constraints for session between join node and system manager

At the end of the joining process, the security manager assigns an initial security level for the session between the joining device and the system manager. That security level shall be 0, 1, 2, 5 or 6; it shall not be 3 (MIC-128) or 7 (ENC-MIC-128), and 4 (ENC-only) is always invalid.

7.4.5 Symmetric-key joining process

7.4.5.1 General

Figure 44 illustrates the messaging involved in the symmetric-key joining process by which a new device shall join an operating network in which it has not recently been a participant. The flow shows the normal case in which no errors or timeouts occur. The timeouts are specified in Table 92.

On the joining device, the symmetric-key joining process shall be initiated with a DMO.Proxy_Security_Sym_Join().Request and finalized with a valid PSMO.Security_Confirm().Response. On the security manager, the symmetric-key joining process shall be initiated with a valid message derived from

PSMO.Security_Sym_Join().Request and finalized with a valid message derived with PSMO.Security_Confirm().Request.

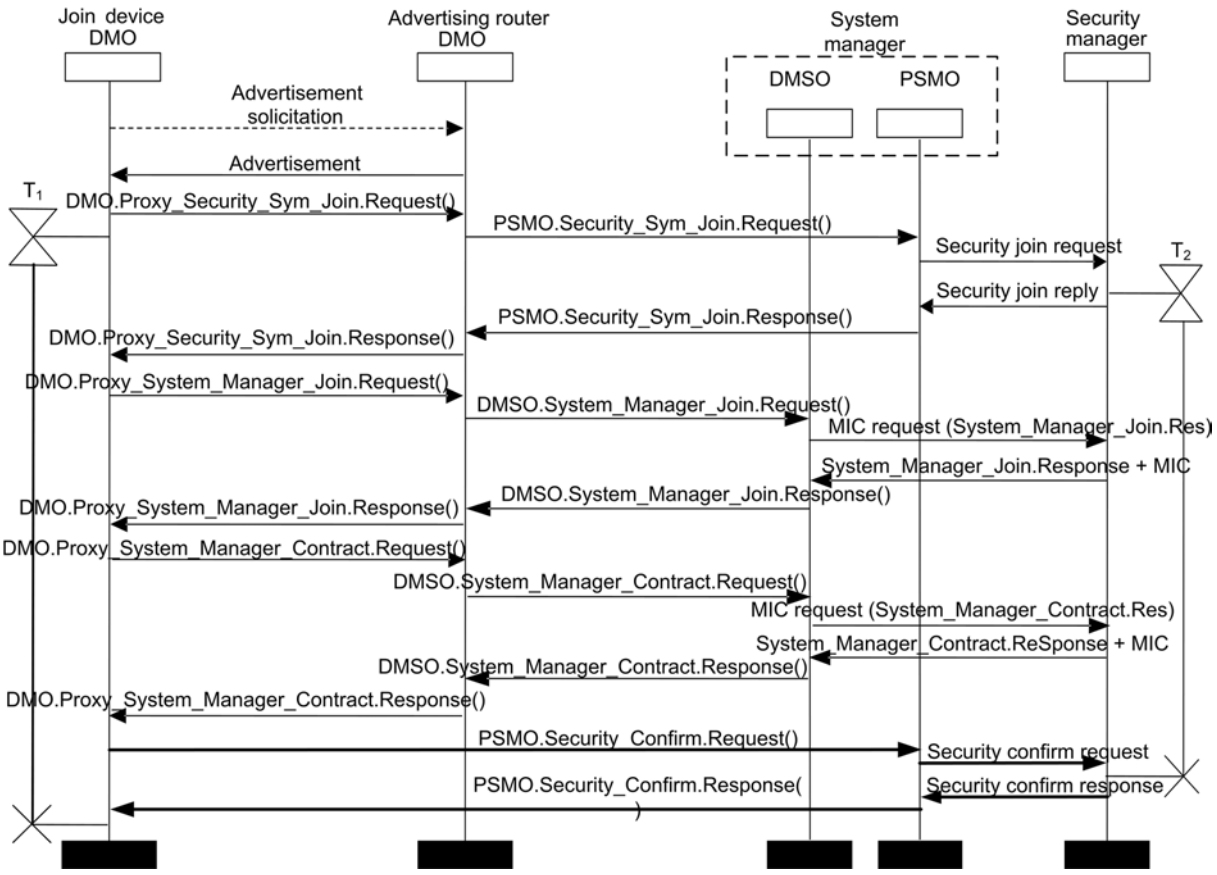


Figure 44 – Example: Overview of the symmetric-key joining process

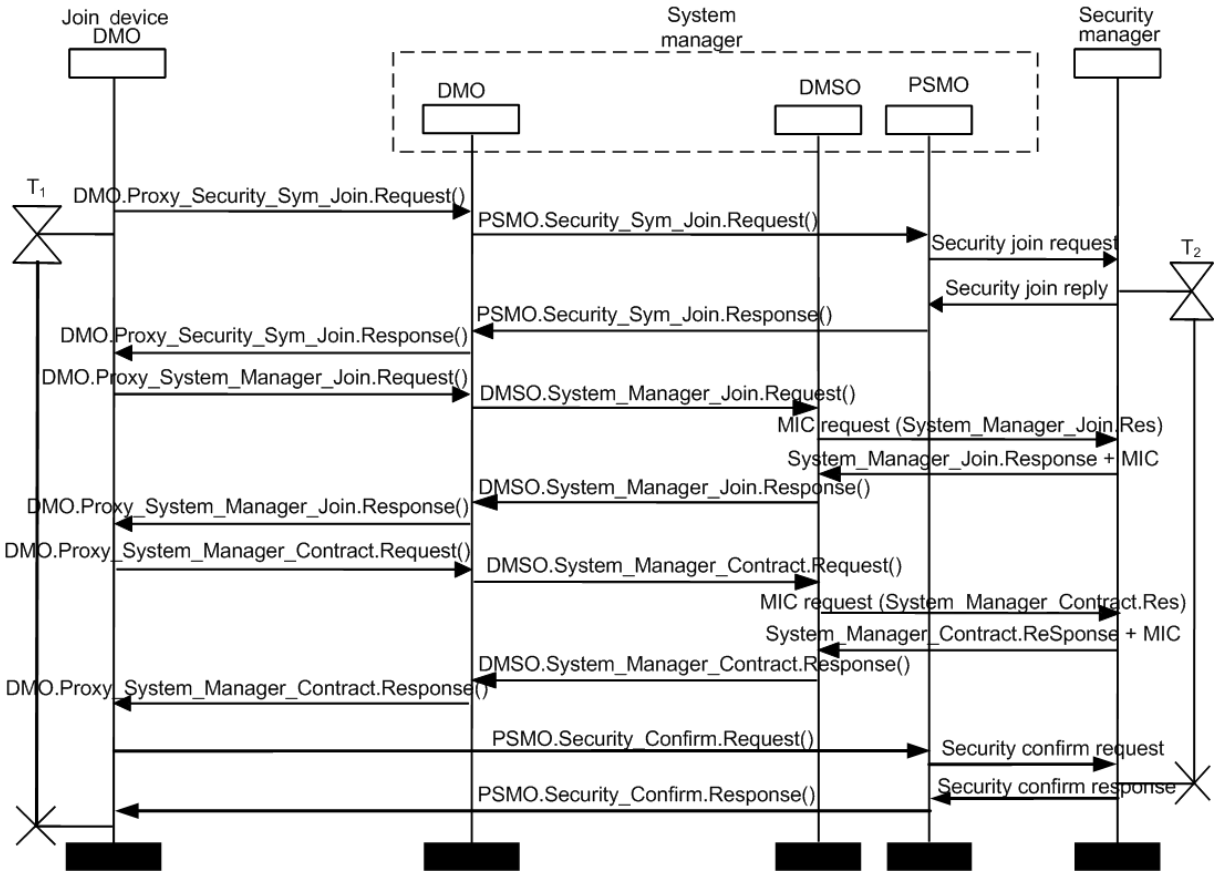


Figure 45 – Example: Overview of the symmetric-key joining process of a backbone device

As shown in Figure 44, a new device shall use the methods defined for the advertising router's DMO to send and receive the join request and join response messages. The methods related to the non-security information are described in 6.3.9.2. The DMO methods related to the security information are described in 7.4.5.2. After transmission of the DMO.Proxy_Security_Sym_Join ().Request, the new device shall start to count join timer JT_1.

The advertising router shall use the methods defined for the system manager's DMSO to send and receive the non-security related join request and response messages. These DMSO methods are described in 6.3.9.5. Methods defined for the system manager's proxy security management object (PSMO) shall be used by the advertising router to send and receive the security related join request and response messages. The PSMO methods related to the security information are described in 7.4.5.2. As shown in Figure 44, the PSMO receives the security-related join request and forwards it to the security manager. The security manager may check a white or black list for the device and ask for human verification before deciding to admit or reject the joining device. If the result is positive, the security manager verifies the cryptographic information of the join request. If the checks fail, the system manager is instructed to revoke the resources allocated to the new device. If the test succeeds, the security manager does the following:

NOTE 1 The methodology of filtering the joining device in the security manager is beyond the scope of this standard.

a) starts join timer JT_2;

b) generates a new master key for the new device;

c) creates a new secure session for the contract between the system manager and the new device;

d) retrieves the current D-key and Crypto Key Identifier for the new device's D-subnet;

e) generates a fresh, unique challenge for the new device;

f) cryptographically protects the aforementioned keys and forms a message integrity check code on the entire response; and

g) sends the security-related response, including the message integrity check code, back to the PSMO.

The PSMO sends this security-related response back to the advertising router which in turn forwards it to the new device.

The new device checks the cryptographic integrity of this security-related response APDU. If the test fails, the received APDU is discarded. If the test succeeds, then the security-related response is processed by the device, which cancels join timer JT_1.

The non-security related join response APDU that is generated by the system manager's DMSO is forwarded to the security manager in order to cryptographically protect the information in the APDU. Once the DMSO receives this protected APDU, it sends the protected APDU back to the advertising router which in turn forwards it to the new device. The new device checks the cryptographic integrity of this protected APDU before using the information in the APDU to complete the joining process. The APDU includes information about the initial contract that the system manager established between the new device and the system manager. This contract is described in 6.3.11.2.6.7.

As part of the last step of the joining process, the new device shall send back a security confirmation APDU to the security manager that contains the challenge from the security manager, authenticated by the new, shared master symmetric key. This security confirmation is sent to the PSMO which forwards it to the security manager. The PSMO method used for this is described in 7.4.5.2.

The security manager checks the confirmation message. If the test fails, the received confirmation response, the join state and the cached information for the new device shall be dropped. If the test succeeds, the security manager cancels its JT_2 timer and sends a confirmation response back to the new device.

If the new device receives a positive response to its confirmation request, it cancels its JT_1 timer.

The contract that was established between the new device and the system manager during the joining process is used to support these messages.

NOTE 2 By sending the response of the challenge authenticated under the master key, the new device proves to the security manager that it was able to extract the master key, and therefore that it had the join key.

The ASL may concatenate ASDUs resulting from multiple method calls into a single TSDU, thus guaranteeing that if one is received, all are received. For example, to reduce the traffic overhead or the join time, the Proxy_Security_Sym_Join().Request and the Proxy_System_Manager_Contract().Request may be concatenated in the same TSDU sent to the advertising router's DMO.

7.4.5.2 Device management object and proxy service management object methods related to the symmetric-key joining process

7.4.5.2.1 General

The new device shall use the Proxy_Security_Sym_Join method defined for the advertising router's DMO in the advertising router to send its security information that is part of the join request and to get its security information that is part of the join response.

NOTE 1 To mitigate flooding by join messages, the system manager limits wireless resources (e.g., timeslots) assigned in advertising routers for receiving joining messages. The resources are described in 9.3.5.2.4.2.

Table 59 describes the Proxy_Security_Sym_Join method. The source object for invoking the DMO.Proxy_Security_Sym_Join().Request shall be the DMO in the Joining device's DMAP.

Table 59 – Proxy_Security_Sym_Join method

Standard object type name: Device management object (DMO)				
Standard object type identifier: 127				
Method name	**Method ID**	**Method description**		
Proxy_Security_Sym_Join	5	Method to use the advertising router as proxy to send a security join request and get a security join response		
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Join_ Request	Security_Sym_Join_Request; see 7.4.5.2.2	Security join request based on symmetric keys from new device that needs to be forwarded to security manager
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Join_ Response	Security_Sym_Join_Response; see 7.4.5.2.3	Security join response based on symmetric keys from security manager that needs to be forwarded to new device; this is protected using the join key

The advertising router shall use the Security_Sym_Join method defined for the system manager's PSMO for sending the security information that is part of the join request on behalf of the new device and to get the security information that is part of the join response.

Table 60 describes the Security_Sym_Join method.

Table 60 – Security_Sym_Join method

Standard object type name: Proxy security management object (PSMO)				
Standard object type identifier: 105				
Method name	**Method ID**	**Method description**		
Security_Sym_Join	1	Method to use the PSMO in the system manager to send a security join request and get a security join response		
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Join_Request	Security_Sym_Join_Request; see 7.4.5.2.2	Security join request from new device to security manager
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Join_Response	Security_Sym_Join_Response; see 7.4.5.2.3	Security join response from security manager to new device that is protected using the join key

As part of the last step of the joining process, the new device shall use the Security_Confirm method defined for the system manager's PSMO for sending a security confirmation to the security manager.

Table 61 describes the Security_Confirm method.

Table 61 – Security_Confirm method

Standard object type name: Proxy security management object (PSMO)				
Standard object type identifier: 105				
Method name	**Method ID**	**Method description**		
Security_Confirm	2	Method used by the new device to send a security confirmation to the security manager through the PSMO		
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Security_Sym_Confirm	Security_Sym_Confirm; see 7.4.5.2.4	Security confirmation from new device to security manager
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	—	—	—	—

NOTE 2 Although the Security_Confirm method does not have any output arguments, the Execution response message in the Application Sublayer is returned as a result of this method.

7.4.5.2.2 Symmetric-key join request

The Security_Sym_Join_Request data structure that is used to form the symmetric-key join request is defined in Table 62.

Table 62 – Security_Sym_Join_Request data structure

Standard data type name: Security_Sym_Join_Request		
Standard data type code: 410		
Element name	**Element identifier**	**Element type**
New_Device_EUI64	1	Type: EUI64Address Classification: Constant Accessibility: Read only
128_Bit_Challenge_From_New_Device	2	Type: SymmetricKey Classification: Static Accessibility: Read/write
Algorithm_Identifier	3	Type: Unsigned8 Classification: Static Accessibility: Read only Default value : 1
MIC	4	Type: Unsigned32 Classification: Static Accessibility: Read only

Fields include:

- New_Device_EUI64 is the EUI64Address of the joining device. This EUI64Address is used by the advertising router when forwarding the message to the system manager to identify this device uniquely, as there could be multiple new devices joining at the same time.
- 128_Bit_Challenge_From_New_Device is a fresh unique challenge generated by the new device to verify that the security manager is alive.
- The algorithm identifier shall be used to specify the symmetric-key algorithm used in the target network. The value of 0x0 shall be reserved. A symmetric-key algorithm of 0x01 corresponding to AES_CCM* shall be the only symmetric algorithm and mode supported for the joining process.

NOTE Currently, only AES_CCM* is defined as a symmetric-key algorithm. However, this field is prepared for algorithms for future use or national regulation.

- The MIC-32 is computed over the elements 1 through 4, using the join key and the 13 most significant octets of the challenge as nonce.

7.4.5.2.3 Symmetric-key join response

The Security_Sym_Join_Response data structure that is used to form the symmetric-key join response is defined in Table 63.

Table 63 – Security_Sym_Join_Response data structure

Standard data type name: Security_Sym_Join_Response		
Standard data type code: 411		
Element name	**Element identifier**	**Element type**
128_Bit_Challenge_From_SecurityManager	1	Type: SymmetricKey Classification: Static Accessibility: Read/write
128_Bit_Response_To_New_Device_Hash_B	2	Type: SymmetricKey Classification: Static Accessibility: Read/write
Combined_Security_Level	3	Type: Unsigned8 (see Table 64) Classification: Static Accessibility: Read/write
Master_Key_HardLifeSpan	4	Type: Unsigned16 Classification: Static Accessibility: Read/write
DL_Key_HardLifeSpan	5	Type: Unsigned16 Classification: Static Accessibility: Read/write
Sys_Mgr_Session_Key_HardLifeSpan	6	Type: Unsigned16 Classification: Static Accessibility: Read/write
DL_Key_ID	7	Type: Unsigned8 Classification: Static Accessibility: Read/write
Encrypted_DL_Key	8	Type: SymmetricKey Classification: Static Accessibility: Read/write
Encrypted_Sys_Mgr_Session_Key	9	Type: SymmetricKey Classification: Static Accessibility: Read/write

This data structure consists of a plaintext section and an encrypted section. The plaintext section shall be composed of the header, the original challenge from the new device, and a new challenge from the security manager which is different from the challenge generated by the new device, and the key policies. The encrypted section shall be composed of the D-key and the T-key with the system manager.

- 128_Bit_Challenge_From_Security_Manager is a fresh unique challenge generated by the Security Manager to verify that the new device is alive.
- 128_Bit_Response_To_New_Device_Hash_B shall be calculated as:
  - Hash_B= HMAC-MMO$_{K_join}$[challenge_from_security_manager || challenge_from_new_device || EUI64Address of new_device || EUI64Address of security_manager || Message_Key_Transport]
  - Message_Key_Transport = Combined_Security_Level || Master_Key_HardLifeSpan || DL_Key_HardLifeSpan || Sys_Mgr_Session_Key_HardLifetime || DL_Key_ID || Encrypted D-key || Encrypted SysMan T-key

- The Master_Key_HardLifeSpan, DL_Key_HardLifeSpan and Sys_Mgr_Session_Key_HardLifeSpan shall be the HardLifeSpan, in units of hours. The Key Type='001' and Key Usage can be inferred implicitly from Table 89 and Table 90 by the element Identifier. A default granularity of 0x2='hours' shall be used for the policies in the join response message.
- The DL_Key_ID shall be the Crypto Key Identifier associated with the D-key sent in the join response. The Crypto Key Identifier of the master key and T-key shall be set implicitly (not transmitted but inferred) as 0x00.
- The 13 most significant octets of the challenge sent from the security manager shall be used as the nonce to encrypt the D-key and T-key. The D-key and T-key are encrypted in the same time (single operation of AES-CCM* encryption with MIC size = 0).
- The response to new device shall be the keyed hash defined as follows:
- The new master key shall be derived as:
  - K_master = HMAC-MMO$_{K_join}$[EUI64Addressnew_device || EUI64Address of security_manager || challenge_from_new_device || challenge_from_security_manager]
- The D-key and the T-key to support the contract with the system manager shall be encrypted using the new master key.

NOTE 1 By including the challenge from the new device and calculating a MIC over it, the security manager proves that it is a live device with knowledge of the join key.

NOTE 2 16 bits of validity period, with units of hours, give a range of over 7 years, which is adequate to express the current maximum key lifetime.

Table 64 – Structure of compressed security level field

Octet	Bits							
	7	6	5	4	3	2	1	0
1	DL_Security_Level			Sys_Mgr_Ses_Security_Level			Master_key_Sec_level	

Fields include:

- DL_Security_Level: The security level applied to the D-key conveyed in the Security_Sym_Join response message. The format of this field shall be as specified in Table 35. Security level 0, None, and security level 4, ENC, shall not be used.
- Sys_Mgr_Ses_Security_Level: The security level applied to the T-key with the system manager conveyed in the Security_Sym_Join response message. The format of this field shall be as specified in Table 35. Security level 4, ENC, shall not be used.
- Master_Key_Sec_Level: The MIC size applied to the master key generated in the joining process. The format of this field is defined in Table 65. Since the encryption factor is different in each message protected by the master key, only the MIC size is specified in this field. The actual security level shall be selected from Table 65 with a combination of encryption conditions in each message.

NOTE 3 For example, since the Security_New_Session_Request and the Security_New_Session_Response data structure don't have any elements to be encrypted, the Security_Level in the Security_Control field is set to MIC-*n* with the MIC size specified in this structure. While the Security_Key_and_Policies data structure has elements to be encrypted, the Security_Level in the Security_Control field is set to ENC-MIC-*n* with MIC size specified in this structure.

Table 65 – Master key security level

Security level identifier	Master_Key_Sec_Level	Security attributes
0	0	Reserved
1	1	MIC-32
2	2	MIC-64
3	3	MIC-128

NOTE 4 Since a MIC is always used to protect the joining process PDUs with the master key, the security level identifier 0 is Reserved.

- ValidNotBefore = TAI time of APDU reception.

NOTE 5 ValidNotBefore can be inferred as the reconstructed time used in the authentication of the PDU, and is not included due to space restrictions in the response PDU.

The time that the APDU is received shall not be more than DSMO.pduMaxAge seconds after it was created. That gives an acceptable start time.

- HardLifeSpan: The key validity duration in hours. A value of 0x0000 shall prohibit the key from expiring.

If HardLifeSpan is zero (i.e., effectively infinite), the inferred key lifetime shall be:

– ValidNotAfter = 0xFFFF FFFF, which is interpreted by key expiration logic as a key that never expires;

– SoftExpirationTime = ValidNotAfter.

If HardLifeSpan is non-zero (i.e., finite), the inferred key lifetime shall be:

– ValidNotAfter = ValidNotBefore + (HardLifeSpan x 3 600);

– SoftExpirationTime = ValidNotBefore + (SoftLifeSpanRatio x HardLifeSpan x 3 600).

A SoftLifeSpanRatio of 50 % shall be used as a default for keys sent with a Key_HardLifeSpan field.

7.4.5.2.4 Symmetric-key security confirmation

The Security_Sym_Confirm data structure that is used to form the symmetric-key security confirmation is defined in Table 66. The source object for invoking PSMO.Security_Sym_Confirm().Request shall be the DMO in the joining device's DMAP.

Table 66 – Security_Sym_Confirm data structure

Standard data type name: Security_Sym_Confirm		
Standard data type code: 412		
Element name	**Element identifier**	**Element type**
128_Bit_Response_To_Security_Manager	1	Type: SymmetricKey Classification: Static Accessibility: Read/write

128_Bit_Response_To_Security_Manager shall be calculated as:

$$\text{HMAC-MMO}_{K\_join}[\text{challenge\_from\_new\_device} \,||\, \text{challenge\_from\_security\_manager} \,||\, \text{EUI64Address of new\_device} \,||\, \text{EUI64Address of security\_manager} \,||\, MIC_1 \,||\, MIC_2]$$

where

MIC_1 is the 32-bit MIC value in the System_Manager_Join response and MIC_2 is the 32-bit MIC value in the System_Manager_Contract response.

NOTE 1 The join confirmation tells the security manager that the device was able to recover the master key using the join key, thus providing proof that the device, which knows the join key, is alive.

NOTE 2 The construction of the hash for the challenge-response protocol was modeled after the protocol outlined in Menezes et al.:1996, 10.17 (see Bibliography).

7.4.6 Asymmetric-key joining process

7.4.6.1 Overview

The asymmetric-key joining process, like the symmetric-key joining process specifies the sequence of steps by which, and the conditions under which, a device may become part of the network and gain access to information required to communicate within the network, both with immediate neighboring devices and with particular infrastructure devices, such as devices assuming the role of system manager or security manager of the network. As such, this entails the sub-processes described below. Note that distribution of the keying material and resource allocation steps are identical for asymmetric-key-based and symmetric-key-based joining processes. The table of roles and their respective bitmap assignment are defined in Annex B.

The enrollment process includes:

- Network membership enrollment. A device and a security manager engage in a mutual entity authentication protocol based on asymmetric-key techniques. This protocol provides evidence regarding the true device identity of both the joining device and the security manager, based on authentic asymmetric keys. In addition, admission may be based on non-cryptographic acceptability criteria (e.g., via a membership test of the device via an access control list). If the device has been positively authenticated and is authorized to join the network, it may be admitted to the network. The entity authentication protocol also results in the establishment of a shared key between the joining device and the security manager, thereby facilitating ongoing secure and authentic communications between these devices.
- Distribution of keying material. A security manager allocates keying material to a newly admitted device, so as to facilitate subsequent communications and continuous authentication of the device to other members of the network as a legitimate network device. The keying material may include D-keys, which are used to evidence network membership amongst devices in the network, and T-keys, which are used to secure and authenticate ongoing communications between a newly admitted device and a system manager.

The joining process assumes that devices have been endowed with sufficient information to allow proper device authentication. A joining device may have been endowed with non-security related information as well.

The asymmetric-key key agreement scheme is specified in 7.4.6.2, the key distribution scheme is specified in detail in 7.4.6.3, the resource allocation scheme is specified in detail in 6.3.9, and the asymmetric-key based join protocol is specified in 7.4.6.3.5. The integers, octets and entities used in the asymmetric-key-based join protocol are defined in Annex F. The asymmetric-key cryptographic building blocks are defined in Annex H.

7.4.6.2 Asymmetric-key key agreement scheme

7.4.6.2.1 Overview

7.4.6.2.1.1 General

Network membership enrollment is based on the execution of the asymmetric-key key agreement scheme specified in H.4.2 and involves device authentication based on implicit certificates, as specified in H.5.1. Both schemes involve asymmetric-key techniques using elliptic curves.

7.4.6.2.1.2 Format of implicit certificate

The implicit certificate is a proof of identity and is used in the asymmetric-key joining process. It can convey an arbitrary data structure; however, to support interworkability among devices, the format of the implicit certificate used in this standard is defined in Table 67.

Table 67 – Implicit certificate format

Element name	Element identifier	Element type
PublicKey_reconstruction_data	1	Type: OctetString37
Subject	2	Type: EUI64Address
Issuer	3	Type: EUI64Address
Usage_serial_number	4	Type: Usage_Serial structure (see Table 68)
ValidNotBefore	5	Type: TAITimeRounded
ValidNotAfter	6	Type: TAITimeRounded

- PublicKey_reconstruction_data: Parameter for generating a public key using the CA's public key.
- Subject: EUI64Address of a device whose public/private key is associated with PublicKey_reconstruction_data
- Issuer: EUI64Address of a device that has generated this certificate.
- Usage_serial_number: Indicating a certification usage and serial number.
- ValidNotBefore: Absolute TAI time (in second) when this certificate becomes valid.
- ValidNotAfter: Absolute TAI time (in second) when this certificate becomes invalid.

Table 68 – Usage_serial_number structure

Octet	Bits							
	7	6	5	4	3	2	1	0
1	Reserved	Issuable	Serial_number					

- Reserved: Reserved field should be 0.
- Issuable: If this field is 0, the key pair corresponding to this certificate shall not be used to sign another certificate. Otherwise, the key pair may be used to sign another certificate.
- Serial_number: Serial number of this certificate managed by the issuer.

7.4.6.2.2 Description of the scheme

Figure 46 illustrates the messaging involved in the asymmetric-key key agreement scheme used with this standard.

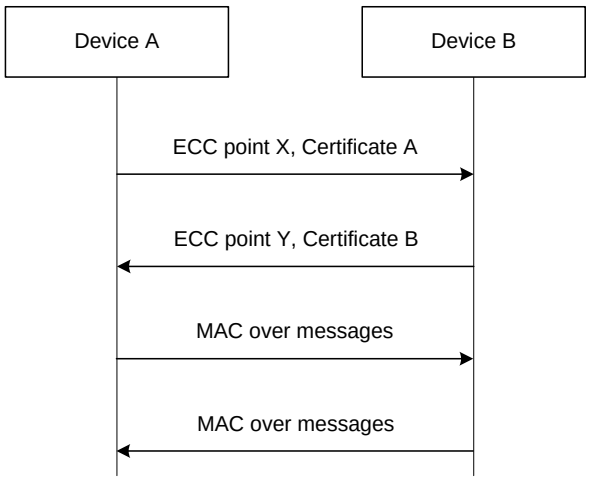


Figure 46 – Asymmetric-key-authenticated key agreement scheme

In the context of the join protocol, the key agreement scheme involves messaging between a joining device and a security manager, whereby the joining device initiates the protocol and whereby the security manager acts as the so-called responder. Thus, in terms of Figure 46, the joining device assumes the role of device A and the security manager assumes the role of device B.

The protocol includes the following sequential components:

a) Key contributions. Each party randomly generates a short-term (ephemeral) public key pair and communicates the ephemeral public key (but not the private key) to the other party. In addition, each party communicates the certificate of its long-term (static) public key to the other party.

b) Key establishment. Each party computes the shared key based on the static and ephemeral elliptic curve points it received from the other party, and also based on the static and ephemeral private keys it generated itself. Due to the properties of elliptic curves, either party arrives at the same shared key.

c) Key authentication. Each party verifies the authenticity of the long-term static key of the other party, to obtain evidence that the only party that may be capable of computing the shared key is indeed the perceived communicating party.

d) Key confirmation. Each party computes and communicates a message authentication check value over the strings communicated by the other party, to evidence possession of the shared key to the other party. This confirms to each party the true identity of the other party and proves that the other party successfully computed the shared key. This key confirmation message may authenticate an additional string communicated by the party itself as well. The strings and string operations are defined in Annex F.

The protocol assumes that each party has access to the root key of the certificate authority (CA) that signed the certificate received from the other party.

7.4.6.2.3 Security properties of the scheme

Successful execution of the complete scheme results in security properties, including the following:

- Mutual entity authentication. Each party has assurances as to the true identity of the other party and that that party was alive during the execution of the protocol.
- Mutual implicit key authentication. Each party has assurances that the only party that may have been capable of computing the shared key is indeed the intended communicating party.

- Mutual key confirmation. Each party has evidence that its intended communicating party successfully computed the shared key.
- Perfect forward secrecy. Compromise of the static key does not compromise past shared keys.
- No unilateral key control. Each party has assurance that neither party was able to control or predict the value of the shared key.
- Additional security properties, such as unknown key-share resilience and known-key security. For details, see ANSI X9.63:2011, Table H.2.

The security services provided by each scheme are assured after successful completion of the complete scheme in question (and if the prerequisites of the scheme are satisfied). From the schemes themselves, it is not clear a priori what properties are provided during execution of the protocol steps of the scheme. The security services provided include:

– Processing of random key contributions does not offer any security services, since these messages are independent.

– From the perspective of the joining device A, the protocol is finished after completion of the processing steps resulting from receipt of the key confirmation message MAC_B, whereas from the perspective of the security manager B, the protocol is only finished after completion of the processing steps resulting from receipt of the key confirmation message MAC_A. In particular, security manager B does not have any assurances prior to receipt and processing of the key confirmation message MAC_A. Thus, any actions by B triggered prior to completion of the entire protocol with A are premature, in the sense that these cannot logically be based on any security assurances (as there are none). In contrast, any actions by B triggered after successful completion of the entire protocol with A may be well-founded, in the sense that these may be based on the security services resulting from the completion of the protocol.

NOTE This re-emphasizes the importance of considering the effect of cryptographic schemes in their entirety.

7.4.6.3 Key distribution scheme

7.4.6.3.1 Overview

Key distribution is based on the shared key resulting from the asymmetric-key key agreement scheme executed between the joining device and the security manager, as described in 7.4.6.2.

7.4.6.3.2 Description of the scheme

The mechanism for distribution of keying material from the security manager to the newly joined device and the system manager is the same as that described in the symmetric-key joining process. For details, see 7.4.4.

7.4.6.3.3 Security properties of the scheme

Successful execution of the key distribution scheme results in security properties including the following:

- Secure and authentic transfer of the D-key and associated keying information from the security manager to the newly joined device.
- Secure and authentic transfer of the T-key and associated keying information from the security manager to the newly joined device and to the system manager selected by the security manager.
- In either case, the distributed keying material is generated by the security manager, thereby offering unilateral key control.

7.4.6.3.4 Formats of protocol messaging

The mechanism for distribution of keying material from the security manager to the newly joined device and the system manager is the same as that described in the symmetric-key joining process. For details, see 7.4.4.

7.4.6.3.5 Asymmetric-key-based join protocol

The asymmetric-key-based join protocol can be viewed as a protocol that combines the asymmetric-key key agreement scheme discussed in 7.4.6.2 and the key distribution scheme discussed in 7.4.6.3, the main difference being in the actual organization of messaging in TPDUs.

The asymmetric-key-based join protocol and the symmetric-key-based join protocol only differ in the use of an asymmetric-key key agreement scheme, rather than a symmetric-key key agreement scheme. Thus, all other aspects of the specification of the symmetric-key-based join protocol (see 7.4.4) apply to the asymmetric-key-based join protocol as well.

7.4.6.4 Asymmetric-key joining process messages

7.4.6.4.1 General

Figure 47 and Figure 48 illustrate the messaging involved in the asymmetric-key joining process by which a new device shall join an operating network in which it has not recently been a participant. The flow shows the normal case in which no errors or timeouts occur. The timeouts are specified in 7.4.7.3.

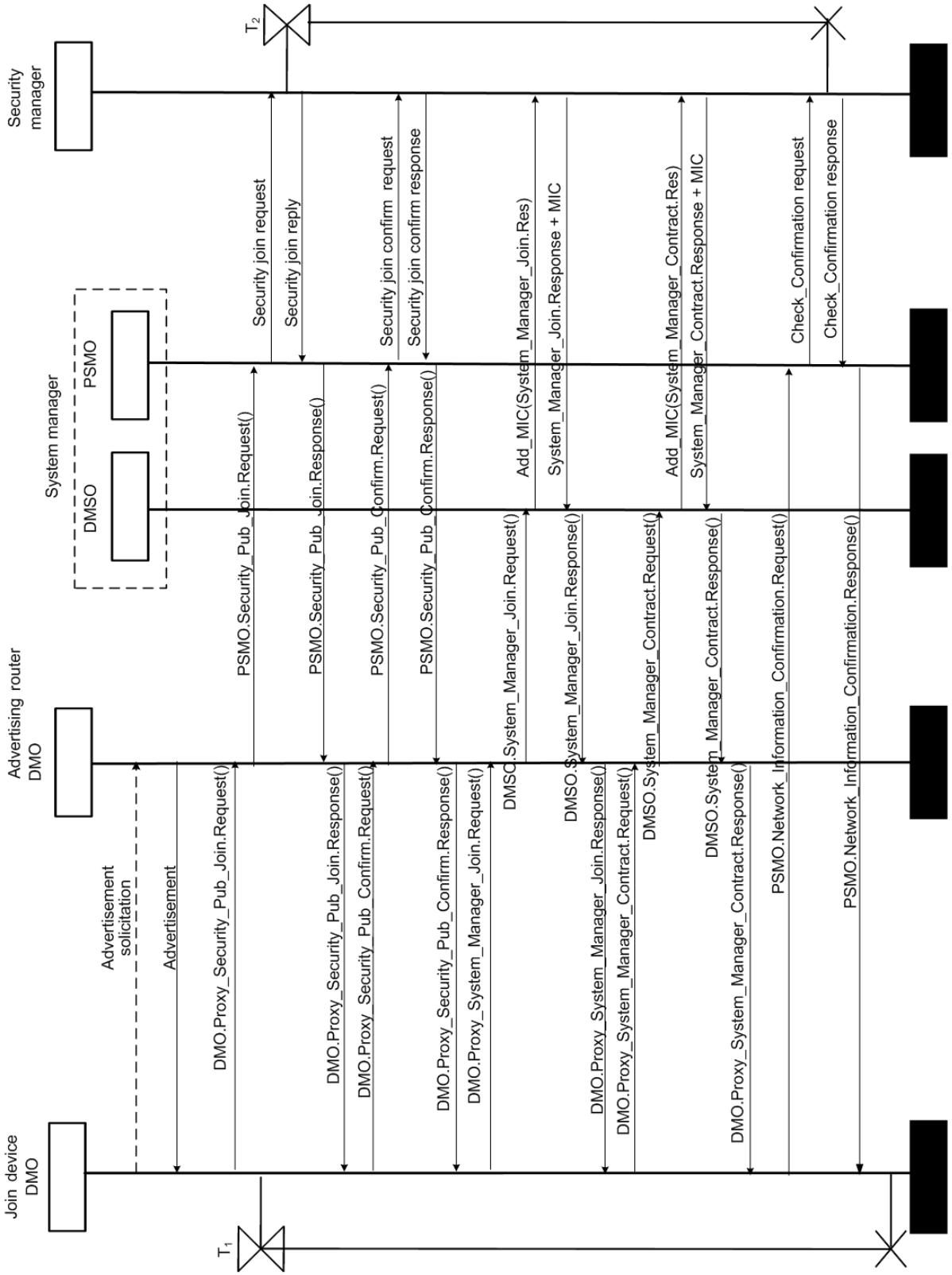


Figure 47 – Example: Overview of the asymmetric-key joining process for a device with a DL

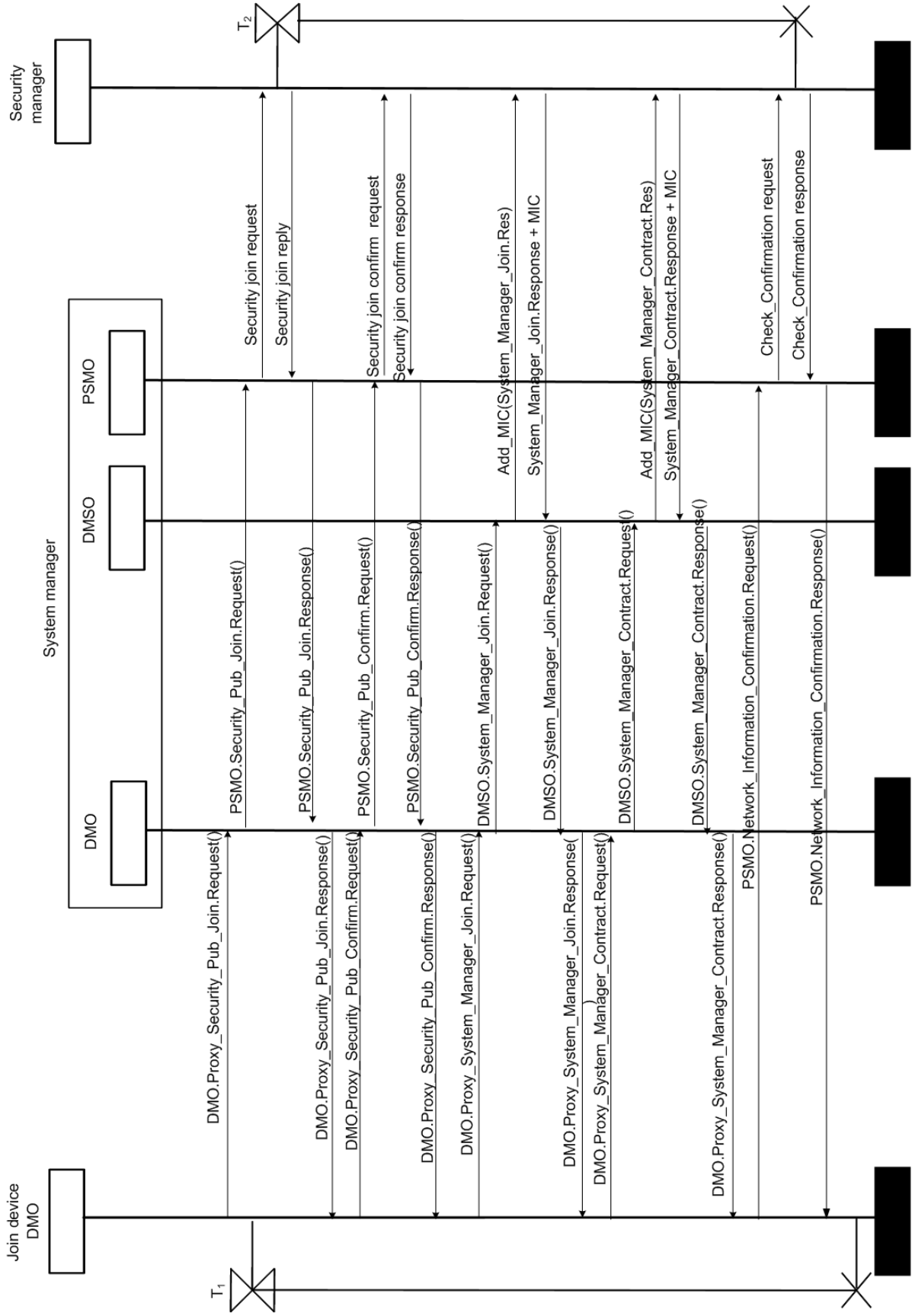


Figure 48 – Example: Overview of the asymmetric-key joining process of a backbone device

On the joining device, the asymmetric-key joining process shall be initiated by transmitting DMO.Proxy_Security_Pub_Join().Request and finalized by receiving a valid PSMO.Network_Information_Confirmation().Response. On the security manager, the asymmetric-key joining process shall be initiated by receiving a valid message derived from PSMO.Security_Pub_Join().Request and finalized by a transmitting message derived to be PSMO.Network_Information_Confirmation().Response.

As shown in Figure 48, a new device shall use the methods defined for the advertising router's DMO to send and receive the join request and join response messages. The methods related to the non-security information are described in 6.3.9.2. The DMO methods related to the security information for the asymmetric join method are described in 7.4.6.4.2.

The advertising router shall use the methods defined for the system manager's DMSO to send and receive the non-security related join request and response messages. These DMSO methods are described in 6.3.9.5. Methods defined for the system manager's proxy security management object (PSMO) shall be used by the advertising router to send and receive the security related join request and response messages. The PSMO methods related to the security information are described in 7.4.5.2.

7.4.6.4.2 Device management object and proxy security management object methods related to the asymmetric-key joining process

The new device shall use the Proxy_Security_Pub_Join method defined for the advertising router's DMO in the advertising router to send its security information that is part of the join request and to get its security information that is part of the join response. After transmitting the DMO.Proxy_Security_Pub_Join().Request, the new device shall start the join timer JT_1. After receiving the DMO.Proxy_Security_Pub_Join().Request, the security manager shall start the join timer JT_2.

Table 69 describes the Proxy_Security_Pub_Join method.

Table 69 – Proxy_Security_Pub_Join method

Standard object type name: Device management object (DMO)				
Standard object type identifier: 127				
Method name	**Method ID**	**Method description**		
Proxy_Security_Pub_Join	6	Method to use the advertising router as proxy to send a security join request and get a security join response		
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Join_Request	Security_Pub_Join_Request; see 7.4.6.4.3	Security join request based on public keys from new device that needs to be forwarded to security manager
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Join_Response	Security_Pub_Join_Response; see 7.4.6.4.3	Security join response based on public keys from security manager that needs to be forwarded to new device; this is protected using the join key

The advertising router shall use the Security_Pub_Join method defined for the system manager's PSMO for sending the security information that is part of the join request on behalf of the new device and to get the security information that is part of the join response.

The source object of the DMO.Proxy_Security_Pub_Join().Request shall be the DMO in the joining device's DMAP.

Table 70 describes the Security_Pub_Join method.

Table 70 – Security_Pub_Join method

Standard object type name: Proxy security management object (PSMO)				
Standard object type identifier: 105				
Method name	**Method ID**	**Method description**		
Security_Pub_Join	3	Method to use the PSMO in the system manager to send a security join request and get a security join response		
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Join_Request	Security_Pub_Join_Request; see 7.4.6.4.3	Security join request from new device to security manager
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Join_Response	Security_Pub_Join_Response; see 7.4.6.4.3	Security join response from security manager to new device that is protected using the join key

After receiving the Proxy_Security_Pub_Join().Response message, the new device shall use the Proxy_Security_Pub_Confirm() method defined for the advertising router's DMO for sending a security confirmation to the advertising router. Table 71 describes this method. The source object of the DMO.Security_Pub_Join().Request shall be the DMO in the joining device's DMAP.

The advertising router shall use the Security_Pub_Confirm method defined for the system manager's PSMO for sending this security confirmation to the security manager. Table 72 describes this method.

The security manager is responsible for checking the confirmation message. If the test fails, the join state and cached information for the new device shall be initialized or dropped. If the test succeeds, then the security manager stops the join timer JT_2, and sends a confirmation response and the non-security information responses to the new device.

If the new device receives a valid response against its confirmation request, then the new device stops the timer JT_1.

Table 71 – Proxy_Security_Pub_Confirm method

Standard object type name: Device management object (DMO)				
Standard object type identifier: 127				
Method name	**Method ID**	**Method description**		
Proxy_Security_Pub_Confirm	7	Method to use the advertising router as proxy by the new device for sending the security confirmation		
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Security_Pub_Confirm_Request	Security_Pub_Confirm_Request; see 7.4.6.4.3	Security confirmation from the new device to the security manager through the advertising router
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Security_Pub_Confirm_Response	Security_Pub_Confirm_Response; see 7.4.6.4.3	Security confirmation from the security manager to the new device through the advertising router

Table 72 – Security_Pub_Confirm method

Standard object type name: Proxy security management object (PSMO)				
Standard object type identifier: 105				
Method name	**Method ID**	**Method description**		
Security_Pub_Confirm	4	Method to send the security confirmation of the new device to the security manager through the PSMO		
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Security_Pub_Confirm_Request	Security_Pub_Confirm_Request; see 7.4.6.4.3	Security confirmation from the new device to the security manager through the advertising router
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Security_Pub_Confirm_Response	Security_Pub_Confirm_Response; see 7.4.6.4.3	Security confirmation from the security manager to the new device through the advertising router

After receiving the DMO.Proxy_System_Manager_Join() and DMO.Proxy_System_Manager_Contract() response, the join device invokes the PSMO.Network_Information_Confirmation() method. Table 73 describes the method.

The Confirm field in PSMO.Network_Information_Confirmation().Request is the MACTag generated with following operation:

$$MACTag = HMAC\text{-}MMO_{K\_join}[MIC_1 \,||\, MIC_2 \,||\, Challenge_{joining\_device}]$$

where:

MIC_1: MIC field in DMO.Proxy_System_Manager_Join().Response (see Table 19);

MIC_2: MIC field in DMO.Proxy_System_Manager_Contract().Response (see Table 20).

Table 73 – Network_Information_Confirmation method

Standard object type name: Proxy security manager object (PSMO)				
Standard object type identifier: 105				
Method name	**Method ID**	**Method description**		
Network_Information_Confirmation	5	Method to make sure that the correct network information was received by the join device		
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Confirm	OctetString16	Confirmation message to make sure the join device received the correct network information from the system manager
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	—	—	—	—

7.4.6.4.3 Formats of protocol messaging

7.4.6.4.3.1 Format of the join request internal structure (PK-join-1)

The Security_Pub_Join_Request data type used in the Security_Pub_Join method and Proxy_Security_Pub_Join method has the following structure and represents the first message flow of the asymmetric-key key agreement scheme (7.4.6.2). This data type is used by the new device and its proxy router in the corresponding methods of the DMO of the proxy router and the PSMO of the system manager respectively. The PK-join-1 data shall be formatted as illustrated in Table 74.

Table 74 – Format of asymmetric join request internal structure

Standard data type name: Security_Pub_Join_Request (PK-Join-1)		
Standard data type code: 415		
Element name	**Element identifier**	**Element type**
New_Device_EUI64	1	Type: EUI64Address Classification: Constant Accessibility: Read only
Protocol control field	2	Type: Unsigned8 Classification: Constant Accessibility: Read only Default value : 1000 0000
Ephemeral elliptic curve point X	3	Type: OctetString37 Classification: Static Accessibility: Read only
Implicit certificate of new device	4	Type: OctetString SIZE(37..66) Classification: Static Accessibility: Read/write
NOTE 1 The format of the implicit certificate used in this standard is defined in 7.4.6.2.1.2. NOTE 2 The total size of the asymmetric-key join request ranges from 83..112 octets. If the user employs this approach, the request sometimes will require more than one conveying DPDU.		

The protocol control field is 1 octet in size and specifies which algorithm is used for the asymmetric-key join protocol and which stage of the protocol is currently being executed. This subfield shall be formatted as specified in Table 75.

Table 75 – Format of the protocol control field

7	6	5	4	3	2	1	0
Algorithm ID = “10”		Reserved				Join subprotocol phase	

The Algorithm ID is 2 bits in size and indicates the asymmetric-key join algorithm in use. The algorithm defined in this standard has ID 0b‘10’.

The join subprotocol phase is 2 bits in size and indicates the current phase of the protocol:

0: Asymmetric-key Join Request;

1: Asymmetric-key Join Response;

2: Asymmetric-key Join Confirm Request;

3: Asymmetric-key Join Confirm Response.

7.4.6.4.3.2 Format of the asymmetric join response internal structure (PK-join-2)

The Security_Pub_Join_Response data type used in the Security_Pub_Join method and Proxy_Security_Pub_Join method has the following structure and represents the first message flow of the asymmetric-key key agreement scheme (7.4.6.2). The PK-join-2 data shall be formatted as illustrated in Table 76.

Table 76 – Format of asymmetric join response internal structure

Standard data type name: Security_Pub_Join_Response (PK-Join-2)		
Standard data type code: 416		
Element name	**Element identifier**	**Element type**
New_Device_EUI64	1	Type: EUI64Address Classification: Constant Accessibility: Read only
Protocol control field	2	Type: Unsigned8 Classification: Constant Accessibility: Read only Default value : 1000 0001
Ephemeral elliptic curve point Y	3	Type: OctetString37 Classification: Static Accessibility: Read only
Implicit certificate of security manager	4	Type: OctetString SIZE(37..66) Classification: Static Accessibility: Read/write
NOTE 1 The format of the implicit certificate used in this standard is defined in 7.4.6.2.1.2. NOTE 2 The total size of the asymmetric-key join request ranges from 83..112 octets. If the user employs this approach, the request sometimes will require more than one conveying DPDU.		

7.4.6.4.3.3 Format of the first join confirmation internal structure (PK-join-3)

The Security_Pub_Confirm_Request data type used in the Security_Pub_Confirm method and Proxy_Security_Pub_Confirm method has the following structure and represents the third message flow of the asymmetric-key key agreement scheme (7.4.6.2). The PK-join-3 data shall be formatted as illustrated in Table 77.

Table 77 – Format of first join confirmation internal structure

Standard data type name: Security_Pub_Confirm_Request (PK-Join-3)		
Standard data type code: 417		
Element name	**Element identifier**	**Element type**
New_Device_EUI64	1	Type: EUI64Address Classification: Constant Accessibility: Read only
Protocol control field	2	Type: Unsigned8 Classification: Constant Accessibility: Read only Default value : 1000 0010
Message_authentication_tag_MAC	3	Type: OctetString16 Classification: Static Accessibility: Read only
Size of text	4	Type: Unsigned8 Classification: Static Accessibility: Read/write Default value : 0 Valid range: 0..31
Text	5	Type: OctetStringN (SIZE : see element 4) Classification: Static Accessibility: Read/write

The Message_authentication_tag_MAC is generated with the following formula:

$$\text{Message\_authentication\_tag\_MAC} = \text{MACmackey}(02_{16} \,||\, U \,||\, V \,||\, QEU \,||\, QEV)$$

where:

U is the EUI64Address of new device;

V is the 8-octet ID of the security manager;

QEU is the octet string of the ephemeral public key of the new device;

QUV is the octet string of the ephemeral public key of the security manager.

This is part of the ECMQV key agreement scheme. The MACmackey is defined in ANSI X9.63:2011, 5.7. In this specification, the keyed hash function HMAC-MMO with the master key shall be used for the MACmackey function.

The text field is used to store any additional information that needs to be authenticated. Users may use this field for any information that requires protection during the asymmetric-key joining process.

7.4.6.4.3.4 Format of the second join confirmation internal structure

The Security_Pub_Confirm_Response data type used in the Security_Pub_Confirm method and Proxy_Security_Pub_Confirm method has the following data structure and represents the fourth message flow of the asymmetric-key key agreement scheme (7.4.6.2). The PK-join-4 data shall be formatted as illustrated in Table 78.

Table 78 – Format of join confirmation response internal structure

Standard data type name: Security_Pub_Confirm_Response (PK-Join-4)		
Standard data type code: 418		
Element name	**Element identifier**	**Element type**
Protocol control field	1	Type: Unsigned8 Classification: Constant Accessibility: Read only Default value : 1000 0011
Message_authentication_tag_MAC	2	Type: OctetString16 Classification: Static Accessibility: Read only
Size of Text	3	Type: Unsigned8 Classification: Static Accessibility: Read/write Default value : 0
Text	4	Type: OctetString (SIZE : see element 3) Classification: Static Accessibility: Read/write
Master_Key_HardLifeSpan	5	Type: Unsigned16 Classification: Static Accessibility: Read/write
Encrypted D-key	6	Type: SymmetricKey Classification: Static Accessibility: Read/write
DL Crypto Key Identifier	7	Type: Unsigned8 Classification: Static Accessibility: Read/write
DL_Key_HardLifeSpan	8	Type: Unsigned16 Classification: Static Accessibility: Read/write
Encrypted system manager T-key	9	Type: SymmetricKey Classification: Static Accessibility: Read/write
System_Manager_Session_Key_HardLifeSpan	10	Type: Unsigned16 Classification: Static Accessibility: Read/write

The Message_authentication_tag_MAC is generated with the following formula:

Message_authentication_tag_MAC = MACmackey(03_{16} || U || V || QEU || QEV)

where:

U is the EUI64Address of new device;

V is the 8-octet ID of the security manager;

QEU is the octet string of the ephemeral public key of the new device;

QUV is the octet string of the ephemeral public key of the security manager.

This is part of ECMQV key agreement scheme. The MACmackey is defined in ANSI X9.63:2011, 5.7. In this specification the HMAC-MMO with the master key shall be used for the MACmackey function.

The text field is used to store user-determined information that needs to be authenticated. Users may use this field for any information to be protected during the asymmetric-key joining process.

The key material and policy fields are the same as for the symmetric-key joining process. Specifically, the following shall be the same as in 7.4.5:

- Master key compressed policy
- Encrypted D-key
- DL Crypto Key Identifier
- D-key compressed policy
- Encrypted system manager T-key
- System manager T-key compressed policy

7.4.7 Joining process and device lifetime failure recovery

7.4.7.1 General

At any point during the joining process, there is a possibility that a PDU will be dropped. In this case, the system should be able to recover and proceed. The following state definition and transition outline the recovery mechanism, along with the triggered side effects.

7.4.7.2 Device states during the joining process and device lifetime

The device states during the joining process are:

- Provisioned: No master key, not in the process of getting the master key.
- Joining: No master key, in the process of getting the master key.
- Joined: Having the current master key, not in the process of getting the next master key.
- Updating: Having the current master key, in the process of getting the next master key.
- Overlapped: Having both the current master key and the next master key.

7.4.7.3 State transitions

The state transitions for a device joining the network shall be as outlined in Table 79 and Figure 49. The timeout values for the joining process join_timeout (ID 4) in Table 92 shall be configurable using DL advertisements. The erasing of keys should be at least equivalent to the clearing of confidential data, as defined in NIST SP 800-88:2012, Table 2-1.

Table 79 – Joining process and device lifetime state machine

Transition	Current state	Event(s)	Action(s)	Next state
T1	Provisioned	DMO initiates the joining process	Advertising router DMO.Proxy_Security_Sym_ Join().Request or DMO.Proxy_Security_Pub_ Join().Request	Joining
T2	Joining	DMO.Proxy_Security_Sym_Join ().Response or DMO.Proxy_Security_Pub_Join ().Response received & crypto check ok	Populate appropriate entries in DSMO and KeyDescriptor Call PSMO.Security_Confirm(). Request (may be delayed), or Call PSMO.Network_Information_ confirmation().Request	Joined
T3	Joined	SoftExpirationTime of master key expired	Call PSMO.Security_New_Session() .Request with the security manager	Updating
T4	Updating	DSMO.New_Key().Request(master_key) from security manager via the PSMO and crypto check ok	Save master key material and policy. Set Key_ID of session to the value assigned by the security manager. Return a DSMO.New_Key().Response	Overlapped keys
T5	Joined	DSMO.New_Key().Request(ma ster_key) from security manager via the PSMO and crypto check ok	Save master key material and policy. Set Key_ID of session to the value assigned by the security manager. Return a DSMO.New_Key().Response	Overlapped keys
T6	Overlapped keys	ValidNotAfter of old master key expired	Remove expired master key	Joined
T7	Updating	Timeout or PSMO.Security_New_Session() .Response&& crypto check ok && SESSION_DENIED	Set the next retry time	Joined
T8	Updating	ValidNotAfter of master key expired	Remove expired master key	Provisioned
T9	Joined	ValidNotAfter of master key expired	Remove expired master key	Provisioned
T10	Joining	Timeout	Reset state machine	Provisioned

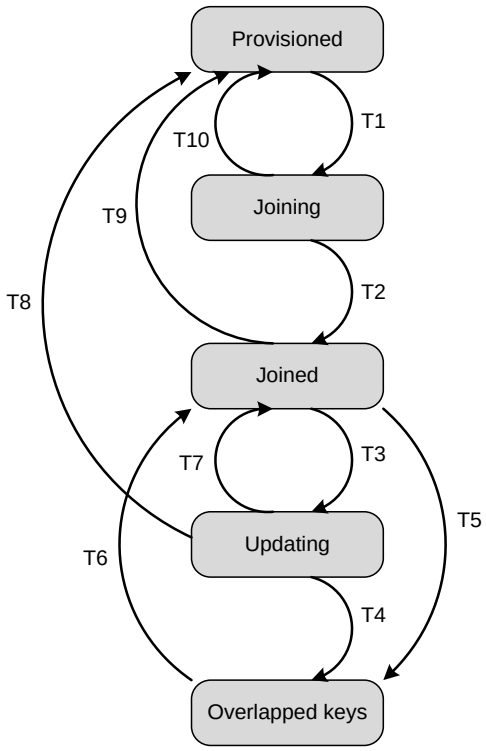


Figure 49 – Device state transitions for joining process and device lifetime

7.5 Session establishment

7.5.1 General

The session establishment occurs in support of an end-to-end secure communication between two UAPs. The end point of a session is defined as the concatenation of the IPv6Address and the transport port. The security manager is responsible for granting or denying the cryptographic material used to establish the end-to-end secure channel between the two devices.

7.5.2 Description

Figure 50 provides a high-level example of session establishment.

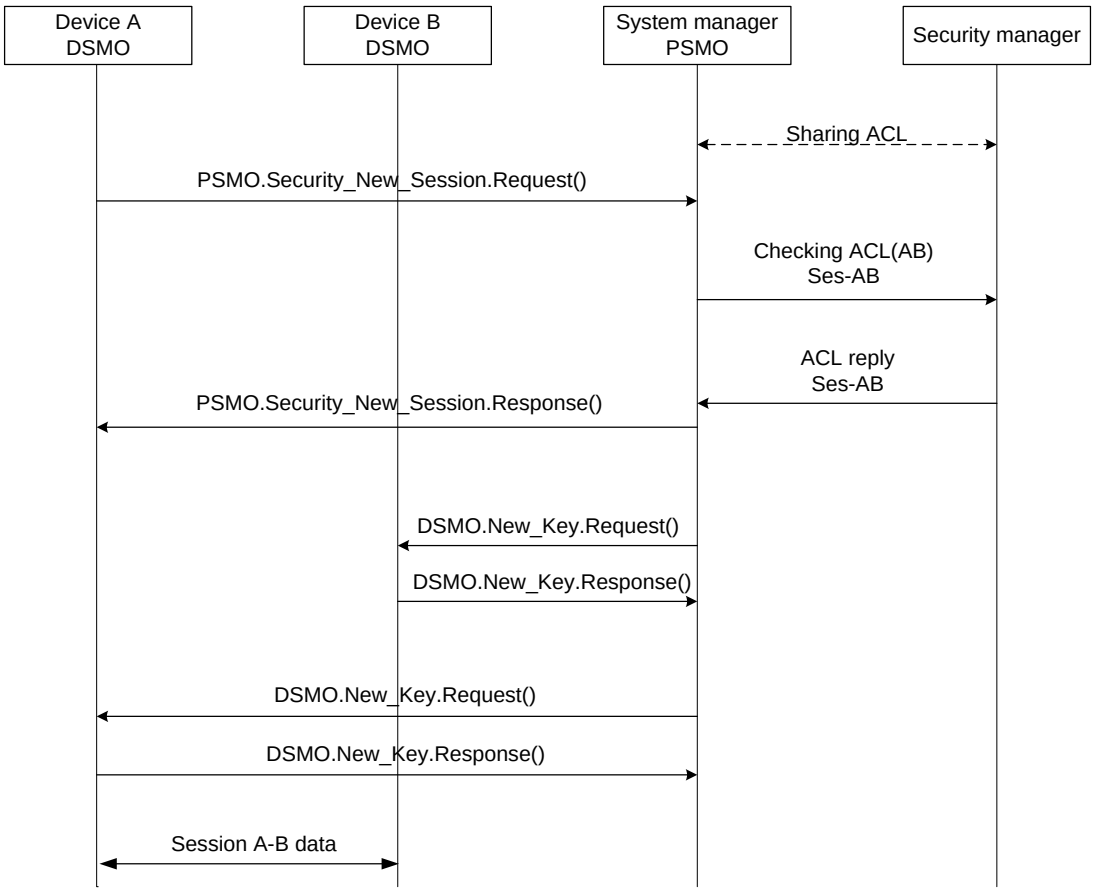


Figure 50 – High-level example of session establishment

In the high-level example shown in Figure 50, a UAP on device A establishes a session with a UAP on device B. The DSMO of device A sends the request to the security manager via the PSMO object of the system manager. The system manager then forwards the request to the security manager, which authenticates the request and may perform a check to verify if the session is allowed. If the session is granted, the security manager generates a single T-key for both end points, encrypts a copy for device A and another copy for device B, and forwards the messages to the system manager's PSMO. The system manager's PSMO can then send the response to the session request. The system manager's PSMO then calls the DSMO method to add a new T-key on device A and device B.

The session establishment may be initiated by a field device or by the security/system manager.

- The field device initiates a session establishment using the PSMO.Security_New_Session() method in Table 80.
- The system/Security manager initiates a session establishment using the DSMO.New_Key() method in Table 83.

The security manager shall assign the same key and Crypto Key Identifier among devices which participate in the secure session. In the case of overlapped keys, the security header needs to convey the Crypto Key Identifier of the key selected to protect the PDU. At the receiver, the device looks for the KeyDescriptor with the Crypto Key Identifier specified in the incoming PDU security header. If the Crypto Key Identifier value does not have a match, the receiving device will not be able to decrypt and/or authenticate the incoming PDU.

7.5.3 Application protocol data unit protection using the master key

7.5.3.1 General

The request is made from the device's DSMO to the system manager's PSMO acting as a proxy to the security manager. The APDU shall be protected with using almost the same PDU security mechanism as the TL. The cryptographic key shall be the master key, and the nonce shall be constructed in the same manner as the TL in 7.3.3.7, but the TAITimeRounded value shall be used for the Nominal TAI creation time field. See Table 363 for coding rules applied to TAITimeRounded values.

NOTE 1 Since the joining process is done at the AL, there is no security at the TL during that process, except confirmation messages.

Since the granularity of the Time_Stamp field is in seconds, two cryptographic operations using the master key shall not be permitted within the same second.

NOTE 2 If the message rate using the master key exceeds the rate of once per second, there will be a nonce collision.

7.5.3.2 Replay protection for application protocol data unit protected with the master key

Upon reception of the APDU protected with the master key, the security procedure shall check for any nonce duplicates with a valid MIC in the nonce cache with the corresponding KeyDescriptor. If a duplicate nonce is detected, the procedure shall discard the PDU before processing the ASDU, otherwise the procedure shall store the nonce into the nonce cache in the KeyDescriptor.

7.5.4 Proxy security management object methods related to the session establishment

Table 80 describes the Security_New_Session method.

Table 80 – Security_New_Session method

<table>
<tr><th colspan="5">Standard object type name: Proxy security management object (PSMO)</th></tr>
<tr><th colspan="5">Standard object type identifier: 105</th></tr>
<tr><th>Method name</th><th>Method ID</th><th colspan="3">Method description</th></tr>
<tr><td rowspan="8">Security_New_Session</td><td>6</td><td colspan="3">Method to use the PSMO in the system manager to send a security session request and get a security session response</td></tr>
<tr><td colspan="4">Input arguments</td></tr>
<tr><td>Argument number</td><td>Argument name</td><td>Argument type (data type and size)</td><td>Argument description</td></tr>
<tr><td>1</td><td>New_Session_Request</td><td>Security_New_Session_Request; see Table 81</td><td>Security new session request from a device to the security manager</td></tr>
<tr><td colspan="4">Output arguments</td></tr>
<tr><td>Argument number</td><td>Argument name</td><td>Argument type (data type and size)</td><td>Argument description</td></tr>
<tr><td>1</td><td>New_Session_Response</td><td>Security_New_Session_Response; see Table 82</td><td>Security new session response from the security manager to the requesting device, protected using the master key</td></tr>
</table>

The Security_New_Session_Request data structure that is used to form the session request is defined in Table 81.

Table 81 – Security_New_Session_Request data structure

Standard data type name: Security_New_Session_Request		
Standard data type code: 420		
Element name	**Element identifier**	**Element type**
Originator_IPv6Address	1	Type: IPv6Address Classification: Static Accessibility: Read/write
Originator_Port	2	Type: Unsigned16 Classification: Static Accessibility: Read/write
Destination_IPv6Address	3	Type: IPv6Address Classification: Static Accessibility: Read/write
Destination_Port	4	Type: Unsigned16 Classification: Static Accessibility: Read/write
Algorithm_Identifier	5	Type: Unsigned8 Classification: Static Accessibility: Read only Default value : 1 = AES_CCM*
Protocol_Version	6	Type: Unsigned8 Classification: Static Accessibility: Read only Default value : 1 = IEC 62734 Ed. 1.0 (i.e., this standard)
Security_Control	7	Type: Unsigned8 Classification: Static Accessibility: Read/write
Crypto_Key_Identifier	8	Type: Unsigned8 Classification: Static Accessibility: Read/write
Time_Stamp	9	Type: TAITimeRounded Classification: Static Accessibility: Read only
MIC	10	Type: OctetString (SIZE = 4, 8, 16) Classification: Static Accessibility: Read only

This data structure consists of a plaintext section only protected using the master key shared between the requester of the session and the security manager. The EUI64Address of the requester shall be used in the nonce construction to protect this structure.

- Originator_IPv6Address shall be the IPv6Address of the first end device (usually the source) in the session.

- Originator_Port shall be the T-port of the first end UAP (usually the source) in the session.
- Destination_IPv6Address shall be the IPv6Address of the second end device (usually the destination) in the session.
- Destination_Port shall be the T-port of the second end UAP (usually the destination) in the session.
- Algorithm_Identifier defines the algorithm and mode of operation supported in this session. In the current release this shall be set to 0x1 = AES_CCM*.
- The protocol version identifies the protocol used for this security association. In this standard, this octet shall be 0x01.
- Security_Control shall be as defined in 7.3.1.2. The security level is chosen from MIC-32, MIC-64 and MIC-128 with the master key security level assigned in the joining process. The Crypto Key Identifier Mode shall be '01' corresponding to a Crypto_Key_Identifier Field size of 1 octet.
- Crypto_Key_Identifier shall be the Crypto_Key_Identifier of the master key used in protecting this structure.
- Time_Stamp shall be the full 32-bit truncated representation of TAI time used in the T-nonce construction.
- MIC shall be the integrity code generated by the AES_CCM* computation. The size of the MIC is assigned in Security_Control field.

The nonce used to generate the MIC is formed as outlined in Table 57 with:

– EUI64Address: EUI64Address of the device transmitting the Security_New_Session Request message.

– Nominal TAI time: The Time_Stamp field in the Security_New_Session Request message.

The Security_New_Session_Response data structure that is used to form the new session response is defined in Table 82.

Table 82 – Security_New_Session_Response data structure

Standard data type name: Security_New_Session_Response		
Standard data type code: 421		
Element name	**Element identifier**	**Element type**
Status	1	Type: Unsigned8 Classification: Static Accessibility: Read/write
Security_Control	2	Type: Unsigned8 Classification: Static Accessibility: Read/write
Crypto_Key_Identifier	3	Type: Unsigned8 Classification: Static Accessibility: Read/write
Time_Stamp	4	Type: TAITimeRounded Classification: Static Accessibility: Read only
MIC	5	Type: OctetString (SIZE = 4, 8, 16) Classification: Static Accessibility: Read only

Fields include:

- Status shall be the status of the session where 0x1 = SECURITY_SESSION_GRANTED and 0x0 = SECURITY_SESSION_DENIED.
- Security_Control shall be as defined in 7.3.1.2. The security level is chosen from MIC-32, MIC-64 and MIC-128 with the master key security level assigned during the joining process. The Crypto Key Identifier Mode shall be '01' corresponding to a Crypto Key Identifier field size of 1 octet.
- Crypto_Key_Identifier shall be the Crypto_Key_Identifier of the master key used in protecting this structure.
- Time_Stamp shall be the 32-bit representation of truncated TAI time used in the nonce construction.
- MIC shall be the integrity code generated by the AES_CCM* computation. The size of the MIC is assigned in the Security_Control field.

The nonce to generate the MIC is formed as outlined in Table 57 with:

– EUI64Address: EUI64Address of the device transmitting Security_New_Session Request message.

– Nominal TAI time: The Time_Stamp field from the Security_New_Session Request message.

If the session is granted, the security manager via the PSMO of the system manager shall call the DSMO New_Key method defined in 7.6.3 to write a new T-key in the devices specified in the session request.

7.6 Key update

7.6.1 General

T-keys have a limited lifetime and are updated periodically to ensure that the session is kept alive. The key update process may be initiated by a device, although it should be pushed from the security manager between the SoftExpirationTime and the HardExpirationTime of a T-key.

7.6.2 Description

The key update process is summarized in Figure 51. A TLE may request that the security manager update a T-key. The security manager will then issue a call to the DSMO of the endpoint TLEs, via the PSMO of the system manager, to update the T-key for those TLEs. Each message is protected under the active master key shared between the security manager and the specific TLE's DSMO.

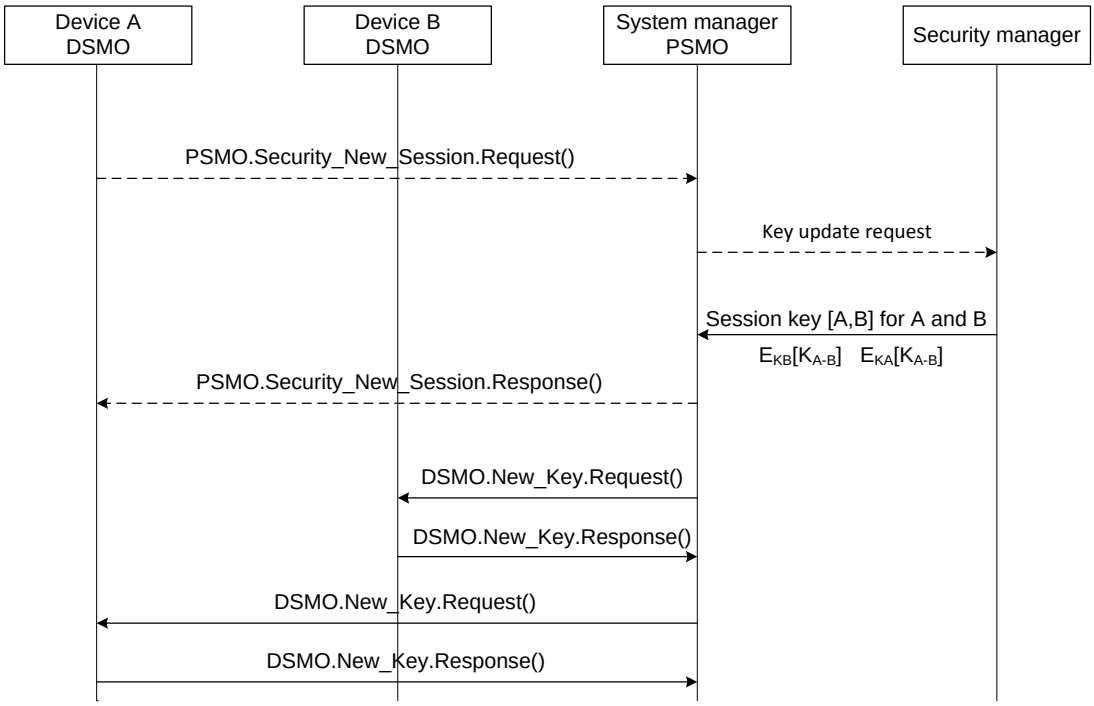


Figure 51 – Key update protocol overview

A TLE participating in a session may initiate the key update process by making a call to the PSMO Security_New_Session method. The request is forwarded from the system manager's PSMO to the security manager, which authenticates the request using the master key of the requesting device. If the check is successful, the security manager recognizes that the session already exists and simply proceeds with the key update protocol exactly as if the SoftExpirationTime of the T-key has expired. The nonce construction for protecting the APDU using the master key is described in 7.5.3.

If the SoftExpirationTime of an active T-key has passed, the security manager shall call the New_Key method on the DSMO of the end devices to write a new key and accompanying policies.

Key Update may also be used to update the DL and master key.

7.6.3 Device security management object methods related to T-key update

Table 83 describes the New_Key method.

Table 83 – New_Key method

Standard object type name: Device security management object (DSMO)				
Standard object type identifier: 125				
Method name	**Method ID**	**Method description**		
New_Key	1	Method to use the DSMO in the device to send a protected security key and accompanying policies		
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Key_And_Policies	Security_Key_and_Policies; see Table 84	Security key and policies to be authenticated, decrypted and stored by a device participating in a session
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Key_Update_Status	Security_Key_Update_Status; see Table 85	Status of the key update, authenticated with the master key

The Security_Key_and_Policies data structure that is used to form the New T-key request is defined in Table 84.

Table 84 – Security_Key_and_Policies data structure

Standard data type name: Security_Key_and_Policies		
Standard data type code: 422		
Element name	**Element identifier**	**Element type**
Key_Policy	1	Type: OctetString (see Table 88) Classification: Static Accessibility: Read only
End_Port_Source (elided for DL or master key)	2	Type: Unsigned16 Classification: Static Accessibility: Read/write
EUI64_remote (elided for DL, EUI64Address of security manager for master key)	3	Type: EUI64Address Classification: Static Accessibility: Read/write
128_Bit_Address_remote (elided for DL or master key)	4	Type: IPv6Address Classification: Static Accessibility: Read/write
End_Port_remote (elided for DL or master key)	5	Type: Unsigned16 Classification: Static Accessibility: Read/write
Algorithm_Identifier	6	Type: Unsigned8 Classification: Static Accessibility: Read only Default value : 1 = AES_CCM*
Security_Control	7	Type: Unsigned8 Classification: Static Accessibility: Read/write
Crypto_Key_Identifier	8	Type: Unsigned8 Classification: Static Accessibility: Read/write
Time_Stamp	9	Type: TAITimeRounded Classification: Static Accessibility: Read only
New_Key_ID	10	Type: Unsigned8 Classification: Static Accessibility: Read only
Key_Material	11	Type: SymmetricKey Classification: Static Accessibility: Read only
MIC	12	Type: OctetString (SIZE = 4, 8, 16) Classification: Static Accessibility: Read only

This data structure consists of a plaintext section only protected using the master key shared between the requester of the session and the security manager. The EUI64Address of the requester shall be used in the nonce construction to protect this structure.

Fields include:

- 128_Bit_Address_remote shall be the IPv6Address of the remote endpoint TLE in this session. In the case of the D-key or master key, this field shall be elided.
- End_Port_remote shall be the T-port of the remote endpoint UAP in this session. In the case of the D-key or master key, this field shall be elided.
- Algorithm_Identifier defines the algorithm and mode of operation supported in this session. In the current release this shall be set to 0x1 = AES_CCM*.
- Security_Control shall be as defined in 7.3.1.2. The security level is chosen from ENC-MIC-32, ENC-MIC-64 and ENC-MIC-128 with the master key security level assigned in the joining process. The Crypto Key Identifier Mode shall be '01' corresponding to a Key Index Field size of 1 octet.
- Crypto_Key_Identifier shall be the Crypto_Key_Identifier of the master key used in protecting this structure.
- Time_Stamp shall be the 32-bit representation of truncated TAI time used in the nonce construction.
- Key_Policy shall be as described in Table 88 and populated by the security manager based on its security policies for this session.
- New_Key_ID shall be the 8-bit Crypto_Key_Identifier assigned to this key material by the security manager.
- Key_Material shall be a symmetric key used for this session.
- MIC shall be the integrity code generated by the AES_CCM* computation. The size of the MIC is specified in the Security_Control field.

The security level for the master key shall be set to at least the strength of the highest key used.

The Security_Key_and_Policies data structure is protected by AES-CCM* with the following parameters:

- authentication part: element 1..10
- encryption part: element 11
- key: master key
- nonce: formed with Table 57 structure with:
  - EUI64Address: EUI64Address of security manager
  - nominal TAI Time: Time Stamp element conveyed in Security_Key_and_Policies

Upon receipt of the DSMO.New_Key().Request method call, the DSMO of the end device shall decrypt and do an integrity check on the PDU using the same incoming PDU processing step as defined in the TL (see 7.3.3.9) with the nonce constructed with the EUI64Address of the security manager and the 32 bits of time included in the PDU. The key used shall be the current master key as identified by the Crypto Key Identifier.

Upon successful completion of the check, the appropriate KeyDescriptor shall be populated using the fields in the Security_Key_and_Policies data structure. In this release, the issuer is always the security manager.

The DSMO shall then generate a status message as defined in Table 85 to notify the security manager of the status of the method call.

The Security_Key_Update_Status data structure that is used to form the response to the key update request is defined in Table 85.

Table 85 – Security_Key_Update_Status data structure

Standard data type name: Security_Key_Update_Status		
Standard data type code: 423		
Element name	**Element identifier**	**Element type**
Status	1	Type: Unsigned8 Classification: Static Accessibility: Read/write
Security_Control	2	Type: Unsigned8 Classification: Static Accessibility: Read/write
Crypto_Key_Identifier	3	Type: Unsigned8 Classification: Static Accessibility: Read/write
Time_Stamp	4	Type: TAITimeRounded Classification: Static Accessibility: Read only
MIC	5	Type: OctetString (SIZE = 4, 8, 16) Classification: Static Accessibility: Read only

Fields include:

- Status shall be the status of the session, where 0x1 = SECURITY_KEY_UPDATE_FAILURE and 0x0 = SECURITY_KEY_UPDATE_SUCCESS.
- Security_Control shall be as defined in 7.3.1.2. The security level is chosen from MIC-32, MIC-64 and MIC-128 with the master key security level assigned during the joining process. The Crypto Key Identifier Mode shall be '01' corresponding to a Crypto Key Identifier Field size of 1 octet.
- Crypto_Key_Identifier shall be the Crypto_Key_Identifier of the master key used in protecting this structure.
- Time_Stamp shall be the 32-bit representation of truncated TAI time used in the nonce construction.
- MIC shall be the integrity code generated by the AES_CCM* computation. The size of the MIC is assigned in the Security_Control field.

The nonce used to generate the MIC is formed as outlined in Table 57 with:

– EUI64Address: EUI64Address of the DLE transmitting the New_Key response message.

– Nominal TAI time: The Time_Stamp element from the Security_Key_Update_Status message.

7.6.4 Failure recovery

7.6.4.1 General

At any point during the session establishment or key update process, there is a possibility that a PDU will be dropped. In this case, the system should be able to recover and proceed. The following state definitions and transitions outline the recovery mechanism, along with the triggered side effects.

7.6.4.2 T-key and D-key states

T-key and D-key states include:

- Idle: No key, not in the process of getting the current T-key.
- Establishing: No key, in the process of getting the current T-key.
- Established: Having the current key, not in the process of getting the next key.
- Updating: Having the current key, in the process of getting the next key.
- Overlapped: Having the current key and the next key

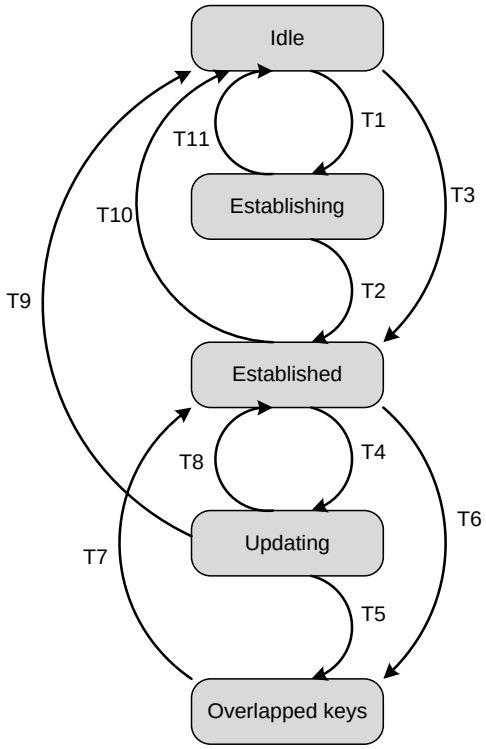
7.6.4.3 T-key and D-key state transition

The state transitions shown in Table 86 and Figure 52 show the state of the device initiating a T-key or D-key retrieval, and of its peer accepting the request. Both start in the idle state, and both end in the established state with a valid session or D-key and relevant cryptographic elements.

Table 86 – T-key and D-key state transition

Transition	Current state	Event(s)	Action(s)	Next state
T1	Idle	DSMO requested new session	Call the following method on the system manager: PSMO.Security_New_Session. Request()	Establishing
T2	Establishing	PSMO.Security_New_Session() .Response && crypto check ok	Save key material, policy and location index, remote addr, remote port, local addr, local port as needed	Established
T3	Idle	DSMO.New_Key().Request from security manager via the PSMO && crypto check ok	Save key material, policy and location index, remote addr, remote port, local addr, local port as needed. Return a DSMO.New_Key.Response()	Established
T4	Established	Session or D-key SoftExpirationTime expired	Call the following method on the system manager: PSMO.Security_New_Session. Request()	Updating
T5	Updating	DSMO.New_Key().Request from security manager via the PSMO && crypto check ok	Save key material, policy and location index, remote addr, remote port, local addr, local port as needed. Return a DSMO.New_Key.Response()	Overlapped keys
T6	Established	DSMO.New_Key().Request from security manager via the PSMO && crypto check ok	Store new keys in memory	Overlapped keys
T7	Overlapped keys	ValidNotAfter of current session or D-key expired	Remove expired key	Established
T8	Updating	Timeout OR PSMO.Security_New_Session. Response() && crypto check ok && SESSION_DENIED	Set time of next retry	Established
T9	Updating	ValidNotAfter of last session or D-key expired	Remove expired key	Idle
T10	Established	ValidNotAfter of last session or D-key expired	Remove expired key	Idle

Transition	Current state	Event(s)	Action(s)	Next state
T11	Establishing	Timeout	Reset state machine and set next retry time if necessary	Idle



NOTE 1 If a device receives the DSMO.New_Key() request while it is in the Updating or Overlapped state, the device will discard the master key, which is not used to encrypt the new master key.

NOTE 2 If the device receives through a DSMO.New_Key() request a new master key which was encrypted using an unknown master key, the device is able to query the security manager for the needed master key for decryption with PSMO.New_Session_Request() request, to re-synchronize the master keys. The security manager has the necessary information to infer, select and use the appropriate master key.

Figure 52 – Device key establishment and key update state transition

7.7 Functionality of the security manager role

7.7.1 Proxy security management object

The attributes of the PSMO are given in Table 87.

Table 87 – Attributes of PSMO in the system manager

Standard object type name: Proxy security management object (PSMO)				
Standard object type identifier: 105				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Reserved for future editions of this standard	1..63	—	—	—

NOTE An object has attributes or methods or both. This object uses only standard methods; no standard attributes have been identified as necessary. Hence in this edition of this standard Table 87 is empty (null).

7.7.2 Authorization of network devices and generation or derivation of initial master keys

The security manager maintains a database containing:

- a list of devices whose credentials have been established through a provisioning process or upon first attempt to join the network, and that have not been revoked; and
- a list of valid join keys and their associated lifetimes that have been issued to provisioning agent devices, which might be provided by new devices that attempt to join the network.

When a new device attempts to join the network and its request comes to the security manager via the system manager's PSMO as described in 7.4.5, the security manager examines the first two lists for a quick accept/reject decision. Otherwise, the procedure shall be as described in 7.4.6.

7.7.3 Interaction with device security management objects

A security manager interacts with a device's DSMO via the PSMO:

- during the joining process;
- when distributing new keys;
- when receiving new keys that have been established by other devices through use of key agreement protocols;
- during a joining process as described in 7.4.4; and
- during key recovery when a new network security manager replaces a failed one.

NOTE Only the security manager generates new keys once a device has joined the network.

7.7.4 Management of operational keys

7.7.4.1 General

The security manager maintains the current master key and associated key-generation and policy attributes for each device it manages.

All symmetric keys are maintained in the security manager's operational storage, which in higher-security implementations should be physically protected within the security manager crypto module.

Where the participating devices do not both implement the set of asymmetric cryptography primitives specified in 7.4.6, which is a construction option for each device, the security manager may also generate shared-secret symmetric data keys for their unicast DL and TL associations.

NOTE Many devices do not have a high-entropy source of random bits. Without such a source, any key component generated by the device is potentially susceptible to inference.

7.7.4.2 Key archiving

Regulation or policy may require that keys be archived to permit concurrent or subsequent decryption of encrypted messaging.

7.7.4.3 Key recovery

The security manager should support two forms of key recovery:

- recovery by a field device that has lost keys that were maintained in volatile storage (e.g., RAM) due to power failure or uncorrected memory error; and
- recovery by a new network security manager of the operational keys currently in use in the network.

Each field device that supports asymmetric-key cryptography shall keep in non-volatile storage:

– its EUI64Address; and

– its public/private key pair, with the latter as a signed certificate if that is available.

Each field device that supports only symmetric-key cryptography shall keep in non-volatile storage:

– its EUI64Address;

– its join key; and

– its current join key and related keying information, if it has previously been a member of the network.

All other operational keying information may be kept in volatile storage, subject to loss upon device power failure or memory corruption, since this information can be regenerated by a security manager once that security manager has determined the EUI64Address of the device.

7.7.4.4 Security policy administration

Primary deployment options affecting network-wide security policy are selected during initial setup of the security manager for a network. Other policy deployment options may be selected at a later time.

7.8 Security policies

7.8.1 Definition of security policy

In this standard, the security policy is defined as a combination of the following parameters:

- Key Type defined in Table 89;
- Key Usage defined in Table 90;
- Key Lifetime defined in 7.2.2.4; and
- Security Level defined in 7.3.1.1.

Keys are distributed with the above parameters specified explicitly or reconstructed implicitly at the recipient. A corresponding KeyDescriptor is generated with those parameters.

7.8.2 Policy extent

Security policies constrain the security choices that individual programs and devices can make. These policies exist at the following levels:

- subnet-wide, across all devices participating in a given D-subnet, which may encompass the entire networked system;
- device-wide, across all application programs and supporting communications layers within the device;
- key-wide, across all PDUs secured with a given key; and
- link-wide, across all PDUs transmitted over a given connection defined by a source and a destination, which may include UAP ports, thus providing UAP-wide policies, across all service invocations by a given application.

Some system-wide policies shall be established before system operation begins; others can be changed dynamically while the system is operating, without interrupting ongoing sessions.

7.8.3 Unconstrained security policy choices

System security policy choices may be made during system operation. The new policy will go into effect with the next rekeying of the affected devices by a security manager. Thus, operation with a given symmetric key always has a fixed set of attributes.

7.8.4 Policy structures

The format of the policies is outlined in Table 88.

Table 88 – Structure of policy field

Octet	Bits							
	7	6	5	4	3	2	1	0
0	Key_Type			Key_Usage			Granularity	
1..4	Nominal ValidNotBefore							
5..6 (7, 8 opt)	HardLifeSpan							
9	Security_Level			Reserved				

Fields include:

- Key_Type: type of key defined in Table 89.
- Key_Usage: usage of key defined in Table 90.
- Granularity: unit in which Nominal HardLifeSpan is interpreted as defined in Table 91.
- Nominal ValidNotBefore in seconds: absolute TAI time in TAITimeRounded form, when the key recipient can start to use the key.
- HardLifeSpan: duration of time for which that key is valid. The key valid duration starts from ValidNotBefore. If the ValidNotAfter field is filled with 0x00 for any granularity, that key has an infinite lifetime and thus the key will never expire. Unless ValidNotAfter is infinite, the actual time duration of the KeyHardLifeSpan shall not exceed 48,5 days (see 7.3.2.4.10) in any granularity.
- Security_Level: security level for each key, as defined in Table 35.
- Reserved: reserved field should be 0.

The possible values for the key types are outlined in Table 89.

Table 89 – Key_Type

Key_type value	Description
0	Reserved
1	Symmetric-key keying material, encrypted
2	ECC manual certificate
3	ECC implicit certificate
4..7	Reserved

The possible values for the key usage are outlined in Table 90.

Table 90 – Key_Usage

Key_Usage value	Description
0	Group key for PDU processing (i.e., D-key)
1	Link key for PDU processing (i.e., T-key)
2	Master key for session establishment
3	Join key
4	Public-key for ECMQV scheme
5	Root key CA for ECQV scheme
6	Reserved
7	Fixed global non-secret key

The granularity of the HardLifeSpan in the key policy is outlined in Table 91.

Table 91 – Granularity

Granularity	SI time unit [a]	Common name	Scale factor	HardLifeSpan (octets)
0	s	Second	1 s	4
1	min	Minute	60 s	3
2	h	Hour	3 600 s	2
3	d	Day	86 400 s	2
[a] Although “s” is the only official SI time unit, the other units listed in the second column are accepted for use with the SI system.				

The following policies shall be available to a network specified by this standard. The variable *k* indicates a variable that may be set by the security manager of a given network.

- Alerts and logging:

  A device keeps track of the number of failed cryptographic computations over a period of time. If a configurable threshold is exceeded, an alert is generated. Alerts include Data DPDU failure rate exceeded, TPDU failure rate exceeded, and key update failure rate exceeded. See alerts in 7.11.4.

- Device policy:

  D-authentication shall always be active with an authentication tag size of 32 bits. The default key used by a joining device is the well-known K_global used to detect random errors only. A secret key protects the higher-level APDU during a secure join. See 7.4.

  NOTE 1 The DL MIC size is specified in 7.3.1.1 and the constraints for the DMIC are specified in 7.3.2.

- Key policy:

  The link, security association and PDU policy are applied through the Key policy. All users of a given key shall have the same policy; see Table 88. The configurable elements include:

  – types of key; see Table 89;
  – granularity of TAI time used in the key lifetime; see Table 91;
  – MIC size (32, 64 or 128 bits); see Table 35;
  – DMIC size of 32, 64 or 128 bits, set to 32 by default, set in the DMXHR; see 7.3.2.2;
  – TMIC size of 0, 32, 64, or 128, set to 0 if security is off, set to 32 by default if security is on. Soft lifetime; see Table 93;
  – payload encryption on/off;

  - DL encryption on/off, set to off by default, set in the DMXHR; see 7.3.2.2;
  - TL encryption on/off, set to off if 'security' is off, set to on if 'security' is on;
  - HardLifeSpan (see Table 88), expressed as an absolute value of TAI time, limited by the maximum duration from the time of key generation that will prevent a rollover of the nonce;

    NOTE 2 This issue is linked to the TAI time granularity in the nonce (see 6.3.10); 48,5 days if used at 1 024 PDU/s.
  - key originator: the EUI64Address of the generator of a given key (usually the security manager) which shall set the policy for a given key;
  - allowed sessions in the security manager.

- Access control policy:

  The access control function for a session establishment is required only in the security manager. The security manager decides to grant or deny a session in the session establishment phase. The result is returned by the PSMO.Security_New_Session() method. If a security manager has both an Allowed and a Disallowed list, the security manager may indicate which one has precedence.

  - Allowed list: The security manager may have a list of allowed devices identified by valid information (e.g., EUI64Address and TAG name) listed in Table 372.
  - Disallowed list: The security manager may have a list of disallowed devices identified by valid information (e.g., EUI64Address and TAG name).

7.9 Security functions available to the AL

7.9.1 Parameters on transport service requests that relate to security

UAPs are permitted to establish application associations dynamically by requesting a session to be established. See 6.3.11.2.5.2. After a session is established, all communications from that UAP with those peers is handled securely until a period of non-use or until a need to reuse storage for security state information causes the TL to terminate the prior transport security association. If a subsequent transport service request from the UAP to those peers occurs after the transport security association has been discontinued, that subsequent request shall be treated as a new request, resulting in a new transport security association.

There is intentionally no ability to carry unsecured TPDUs on the transport security association once it has been established, since such a mechanism would be trivially easy to attack simply by altering selected authenticated TPDUs to indicate that they employed no authentication.

To support stateless AL services, the least-recently-used policy may be applied by underlying layers for recycling any resource commitments (e.g., security connection state) that they might make.

It would assist efficient security system operation if each transport service request on an association had the ability to hint at the expected interval before next use of the association. Such hinting provides guidance to the management of the implicit transport security connections needed for secured transport communications, permitting intelligent caching of established security connections and minimizing the thrashing that occurs when an implicit security connection is closed and then re-opened after a new key is established at all association participants.

The permitted security levels (see 7.3.1.1) on a transport service request are:

- encryption of the TPDU upper-layer payload: on/off;
- authentication of the TPDU with a TMIC of size 32, 64 or 128 bits.

In an API, these may be conveyed jointly as a single signed integer (e.g., an Integer8), where the sign was used to designate encryption (–) or not (+), and the magnitude was used to specify the requested size nn, with the value zero representing a request for no authentication and no encryption.

7.9.2 Direct access to cryptographic primitives

7.9.2.1 General

UAPs may use any of the cryptographic services available to a device. These include:

- unkeyed and keyed hash functions;
- pseudo-random or true-random bit string generation;
- symmetric-key cryptography;
- block cipher encryption;

NOTE 1 Exclusion of block cipher decryption makes it more likely that an implementation is able to use hardware assistance.

- stream cipher functions for processing data strings that include authentication, encryption, extended authentication with encryption, decryption, and decryption with extended authentication.

The available cryptographic primitives also may include a single construction option:

- asymmetric-key cryptography:
  - encryption with a public key and decryption with a private key of a private/public key pair;
  - signing with a private key and signature authentication with a public key of a private/public key pair;
  - key pair generation;
  - certificate generation, signing, and self-signing;
  - two-party Menezes-Qu-Vanstone key agreement;
  - Pintsov-Vanstone digital signatures.

NOTE 2 The single asymmetric-key-cryptography construction option provides all of these capabilities.

Abstract service definitions for all of the primitives of 7.9.2 are specified in 6.2.3.

7.9.2.2 Unkeyed hash functions

A secure unkeyed (or fixed-key) one-way hash function shall be provided.

The default unkeyed hash shall be Matyas-Meyer-Oseas (MMO) as specified in ISO/IEC 10118-2, based on the block cipher of 7.9.3.2.

NOTE Use of the MMO algorithm makes it more likely that an implementation is able to use hardware assistance..

Other unkeyed hash functions may be used where needed, either due to national requirement or because a larger output hash size is required for some application or to counter a threat.

An alternate cryptographic algorithm package may be required for US government systems, because MMO is not authorized for US government use. Other governments may have similar policies.

7.9.2.3 Random bits

Each device shall provide a high-quality source of random bits from a deterministic random bit generator. This may be a properly-seeded generator that is compliant with ANSI X9.82 or FIPS 186-3. Where available, the high-entropy source should be a non-deterministic random bit generator.

A high quality source of random bits shall be used in the asymmetric-key join. A properly seeded deterministic random bit generator may be used in generating challenge values in the symmetric-key join.

NOTE 1 Non-deterministic random bit generators are not suitable for direct use due to the inability to prove any statistical properties of such a source other than its non-determinism. Instead, they are used to seed and provide continuing high-entropy input to deterministic random bit generators, whose statistical properties are quantifiable. Certification of the entropy source (as the certification of the security implementation), being a highly specialized function, is best delegated to an accredited entity. NIST SP 800-22 is useful in testing non-deterministic and deterministic random bit generators.

NOTE 2 In the symmetric-key joining process, it is possible to generate a seed by using the block cipher (whose default is AES) to encrypt the TAI time under the join key (i.e., Seed = Encrypt[K_join, TAI]). Such a join key is presumed to arise from a high entropy source, having been generated in the security manager and distributed during the provisioning phase.

7.9.3 Symmetric-key cryptography

7.9.3.1 Keyed hash functions

The default keyed hash shall be HMAC, based on the unkeyed hash of 7.9.2.1 (see FIPS 198).

7.9.3.2 Block cipher encryption and decryption functions

The default block cipher shall be AES-128, which has a 16 B block size and a 16 B key size (see FIPS 197).

Alternate block ciphers may be used with the appropriate algorithm identifier where needed, either due to national requirements or because a larger key size or block size is required for some application or to counter some threat.

7.9.3.3 Stream cipher functions for encryption, decryption, authentication, extended authentication with encryption, and decryption with extended authentication

The security of this system is based in part on the availability of a stream cipher mode of operation of a block cipher that provides encryption/decryption, authentication, or both. When both are provided, the authentication can extend to data that is not included in the encryption/decryption process.

NOTE Encryption/decryption without authentication is avoided within TPDUs and DPDUs because there are a number of published cryptanalytic attacks that apply to all such schemes. However, the encryption-only and decryption-only modes of CCM* are available to UAPs for their use, such as for protection of the data in place.

The default stream cipher mode of operation of the block cipher of 7.9.3.2 shall be CCM* (see ISO/IEC 19772, mechanism 3). CCM* may be used for authentication-only, for extended-authentication-with-encryption, or for decryption-with-extended-authentication.

7.9.3.4 Secret key generation primitive

A secret key generation (SKG) primitive shall be used by the symmetric-key key agreement schemes specified in this standard.

This primitive derives a shared secret value from a challenge owned by an entity U_1 and a challenge owned by an entity U_2 when all the challenges share the same challenge domain

parameters. If the two entities both correctly execute this primitive with corresponding challenges as inputs, the same shared secret value will be produced.

The shared secret value shall be calculated as follows:

- Prerequisites: the prerequisites for the use of the SKG primitive are:
  - each entity shall be bound to a unique identifier (e.g., the EUI64Address of the device). All identifiers shall be bit strings of the same size. Entity U_1's identifier will be denoted by the bit string U_1. Entity U_2's identifier will be denoted by the bit string U_2;
  - a specialized MAC scheme shall have been chosen, with tagging transformation as specified in ANSI X9.63:2011, 5.7.1. The size in bits of the keys used by the specialized MAC scheme is denoted by macKeySize.
- Input: the SKG primitive takes as input:
  - a bit string MacKey of size macKeySize bits to be used as the key of the established specialized MAC scheme;
  - a bit string QEU_1 provided by U_1;
  - a bit string QEU_2 provided by U_2.
- Actions: the following actions are taken:
  - form the bit string consisting of U_1's identifier, U_2's identifier, the bit string QEU_1 corresponding to U_1's challenge, and the bit string QEU_2 corresponding to U_2's challenge.
- MacData = $U_1 \,||\, U_2 \,||\, QEU_1 \,||\, QEU_2$
  - calculate the tag MacTag for MacData under the key MacKey using the tagging transformation of the established specialized MAC scheme:
- MacTag = MACMacKey(MacData)
  - if the tagging transformation outputs invalid, also output invalid and stop;
  - otherwise, set Z=MacTag.
- Output: the bit string Z as the shared secret value.

7.10 Security statistics collection, threat detection, and reporting

Major security-related events logged by the security manager should include:

- authorizations of new devices;
- first joining of new devices to the network; and
- prolonged disappearance of devices from the network, particularly when they are expected to have a stationary presence.

NOTE The required logging of other security events is a potential subject for future standardization.

The following security-related events shall both be logged and alerted:

- MIC failure rates on received DPDUs that appear to be properly-formed, specifying the proper network-ID, that exceed a range specified in attribute 5 of the DSMO;
- MIC failure rates on received TPDUs that exceed a range specified in attribute 6 of the DSMO; and
- any integrity failure detected when unwrapping a wrapped symmetric key that exceeds a range specified in attribute 9 of the DSMO.

7.11 DSMO functionality

7.11.1 General

The device security management object (DSMO) is part of the DMAP and is the local security management application in each device. It is responsible for the agreement and exchange of cryptographic material along with associated policies. It communicates with the DSMO of the security manager via the proxy security manager object (PSMO) of the system manager. Therefore, TL security shall be used to protect the DSMO traffic, except during the joining process which requires alternative special measures.

7.11.2 DSMO attributes

Table 92 describes the DSMO.

Table 92 – DSMO attributes

Standard object type name: Device security management object (DSMO)				
Standard object type identifier: 125				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
DPDU_MIC_Failure_Limit	1	The threshold of DPDU MIC failures per time unit beyond which an alert will be sent to the security manager	Type: Unsigned16 Classification: Static Accessibility: Read/write Default value: 5	The value is reset to 0 after an alert is generated
DPDU_MIC_Failure_Time_Unit	2	The time interval in seconds used to determine the DPDU MIC failure rate	Type: Unsigned16 Classification: Static Accessibility: Read/write Default value: 60 s	
TPDU_MIC_Failure_Limit	3	The threshold of TPDU MIC failures per time unit beyond which an alert will be sent to the security manager	Type: Unsigned16 Classification: Static Accessibility: Read/write Default value: 5 Valid range: > 0	The value is reset to 0 after an alert is generated
TPDU_MIC_Failure_Time_Unit	4	The time interval in seconds used to determine the TPDU MIC failure rate	Type: Unsigned16 Classification: Static Accessibility: Read/write Default value: 5	
DSMO_KEY_Failure_Limit	5	The threshold beyond which an alert will be sent to the security manager	Type: Unsigned16 Classification: Static Accessibility: Read/write Default value: 1	The value is reset to 0 after an alert is generated
DSMO_KEY_Failure_Time_Unit	6	The time interval in hours used to determine the DSMO key failure rate	Type: Unsigned16 Classification: Static Accessibility: Read/write Default value: 1	

Standard object type name: Device security management object (DSMO)				
Standard object type identifier: 125				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Security_DPDU_Fail_Rate_ Exceeded_AlertDescriptor	7	Used to change the priority of the Security_DPDU_Fail_Rate_Exceeded Alert that belongs to the security category. This alert can also be turned on or off	Type: Alert report descriptor Classification: Static Accessibility: Read/write Default value: [FALSE, 6]	See alert definition
Security_TPDU_Fail_Rate_ Exceeded_AlertDescriptor	8	Used to change the priority of the Security_TPDU_Fail_Rate_Exceeded Alert that belongs to the security category. This alert can also be turned on or off	Type: Alert report descriptor Classification: Static Accessibility: Read/write Default value: [FALSE, 6]	See alert definition
Security_Key_Update_Fail_ Rate_Exceeded_ AlertDescriptor	9	Used to change the priority of the Security_Key_Update_Fail_Rate_Exceeded Alert that belongs to the security category. This alert can also be turned on or off	Type: Alert report descriptor Classification: Static Accessibility: Read/write Default value: [FALSE, 6]	See alert definition
pduMaxAge	10	The maximum amount of time in seconds a PDU is allowed to stay in the network. If a PDU is received in a time window exceeding this period, it shall be rejected at the receiver	Type: Unsigned16 Classification: Static Accessibility: Read/write Default value: 510 Valid range: 0..600	Set to 510 s by default

7.11.3 KeyDescriptor

7.11.3.1 General

The information associated with a key is summarized in Table 93.

Table 93 – KeyDescriptor

Element name	Element identifier	Element scalar type
KeyLookupData*	1	Type: OctetString36 Classification: Static Accessibility: Read/write See Table 94
KeyUsage	2	Type: Unsigned8 Classification: Static Accessibility: Read/write Valid range: 0..7 See Table 90
ValidNotBefore	3	Type: TAITimeRounded Classification: Static Accessibility: Read/write
SoftExpirationTime	4	Type: TAITimeRounded Classification: Static Accessibility: Read/write
ValidNotAfter	5	Type: TAITimeRounded Classification: Static Accessibility: Read/write
Issuer	6	Type: IPv6Address or EUI64Address Classification: Static Accessibility: Read/write
CryptoKeyIdentifier	7	Type: Unsigned8 or Unsigned64 Classification: Static Accessibility: Read/write
KeyMaterial	8	Type: OctetString Classification: Static Accessibility: Read/write
SecurityLevel	9	Type: Unsigned8 Classification: Static Accessibility: Read/write Valid range: 0..7 See Table 35
Counter	10	Type: Unsigned8 Classification: Static Accessibility: Read/write
NonceCache	11	Type: OctetString Classification: Dynamic Accessibility: Read/write
MICFailures	13	Type: Unsigned16 Classification: Static Accessibility: Read/write
NOTE * indicates an index field.		

The T-keyLookupData OctetString fields are listed in Table 94.

Table 94 – T-keyLookupData OctetString fields

Field name	Field scalar type
SourceAddress	Type: IPv6Address
SourcePort	Type: Unsigned16
DestinationAddress	Type: IPv6Address
DestinationPort	Type: Unsigned16

NOTE 1 Since the internal representation of the Key Descriptor is not observable, any representation aspects in the following are purely for exposition.

Key Descriptor fields include:

- KeyLookupData:
  - at the TL, used as index to find a key for a given association;
  - at the DL, this field is not used and shall be set to all 0x00;
- KeyUsage: identifies whether the key is usable as a D-key or a T-key or both;

NOTE 2 Using a 2-bit bitmap allows a key to be defined to be used as a DL and a T-key at the same time, if allowed by the key policy.

- ValidNotBefore: time (TAI) at which the key becomes valid;
- SoftExpirationTime: time (TAI) at which an updated key is needed;
- ValidNotAfter: time (TAI) at which the key becomes invalid;
- Issuer: address of the issuer of the key; this can be an IPv6Address or an EUI64Address;
- CryptoKeyIdentifier: Crypto Key Identifier, set by the key issuer, used to distinguish keys when multiple keys are valid concurrently;
- KeyMaterial: key data for encryption/decryption and/or MIC generation;
- SecurityLevel: as described in Table 88;
- Counter: if KeyUsage bit0 is 0 (this key is not a D-key), this field is not used, so is set to 0;
- NonceCache: if KeyUsage bit1 is 0 (this key is not a T-key), this field is not used, so is set to NULL;
- MICFailures: number of MIC authentication failures after which an alarm should be generated.

7.11.3.2 Additional device security management object methods to support key management

Table 95 describes the delete key method. The result of the method invocation is stored into ServiceFeedbackCode in the application sublayer header and returned to the requesting device. The nonce construction for protecting the APDU using a master key is described in 7.5.3.

Table 95 – Delete key method

Standard object type name(s): Device security management object (DSMO)				
Standard object type identifier: 125				
Method name	**Method ID**	**Method description :**		
Delete_key	2	This method is used to delete a symmetric key on a device. This method is evoked by the PSMO of the security manager. The method shall be protected by the current master key shared between the device and the security manager		
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	KeyUsage	Unsigned8	KeyUsage defined in Table 90
	2	Crypto_Key_Identifier	Unsigned8	The Crypto_Key_Identifier used to uniquely identify keys overlapping in validity period
	3	Source_Port	Unsigned16	Source port; if KeyUsage is not 0x01 (i.e. T-key), this field should be elided
	4	Destination_Address	IPv6Address	Destination Address; if KeyUsage is not 0x01 (i.e. T-key), this field should be elided
	5	Destination_Port	Unsigned16	Destination Port; if KeyUsage is not 0x01 (i.e. T-key), this field should be elided
	6	MasterKeyID	Unsigned8	Crypto Key Identifier for the master key used for generating MIC
	7	Time_Stamp	Unsigned32	Time of creating this message in TAITimeRounded form. This argument is time portion of the nonce used for generating MIC to protect this method call
	8	MIC	OctetString (SIZE = 4, 8, 16)	The integrity check using AES_CCM*. The MIC size is chosen from MIC-32, MIC-64 and MIC-128 with master key security level assigned in joining process
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	—	—	—	—

The MIC is generated by an AES-CCM* operation with the following parameters:

- authentication part: element 1..7;
- encryption part: none;
- key: master key, which has Crypto Key Identifier = MasterKeyID;
- nonce: formed Table 57 structure with:
  - EUI64Address: EUI64Address of Security manager;

– nominal TAI time: Time Stamp field conveyed in Delete_Key() request.

The Key_Policy_Update method is described in Table 96. The result of the method invocation is stored into ServiceFeedbackCode in the application sublayer header and returned to the requesting device. The nonce construction for protecting the APDU using a master key is described in 7.5.3.

Table 96 – Key_Policy_Update method

Standard object type name(s): Device security management object (DSMO)				
Standard object type identifier: 125				
Method name	**Method ID**	**Method description**		
Key_Policy_Update	3	This method is used to update a policy associated with a symmetric key on a device. This method is evoked by the PSMO of the security manager. The method shall be protected by the current master key shared between the device and the security manager		
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	KeyUsage	Unsigned8	KeyUsage defined in Table 90
	2	Crypto_Key_Identifier	Unsigned8	The Crypto_Key_Identifier used to uniquely identify keys overlapping in validity period
	3	Source_Port	Unsigned16	Source port; if KeyUsage is not 0x01 (e.g. T-key), this field should be elided
	4	Destination_Address	IPv6Address	Destination Address; if KeyUsage is not 0x01 (e.g. T-key), this field should be elided
	5	Destination_Port	Unsigned16	Destination Port; if KeyUsage is not 0x01 (e.g. T-key), this field should be elided
	6	SoftLifeSpanRatio	Unsigned8	The percentage of the HardLifeSpan beyond which a key update will be initiated
	7	Security_Level	Unsigned8	Security level specified in Table 35
	8	MasterKeyID	Unsigned8	Crypto Key Identifier for the master key used for generating MIC
	9	Time_Stamp	Unsigned32	Time of creating this message in TAITimeRounded form. This argument is the time portion of the nonce used for generating MIC to encrypt and protect this method call
	10	MIC	OctetString (SIZE = 4, 8, 16)	The integrity check using AES_CCM*. The MIC size is chosen from MIC-32, MIC-64 and MIC-128 with master key security level assigned in joining process
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	—	—	—	—

The MIC is generated by an AES-CCM* operation with the following parameters:

- authentication part: element 1..7;
- encryption part: none;
- key: master key, which has Crypto Key Identifier = MasterKeyID;
- nonce: formed Table 57 structure with:
  - EUI64Address: EUI64Address of security manager;
  - nominal TAI time: Time Stamp field conveyed in Key_Policy_Update() request.

The SoftExpirationTime in the Key Descriptor is updated following a successful MIC check on the parameters of this method call. All the parameters shall be concatenated from the first element to the one before the last element (thus excluding the integrity check). The key SoftLifeSpanRatio is the percentage of the difference between the ValidNotAfter and the ValidNotBefore time. For example, a SoftLifeSpanRatio of 50 % would cause a key update half way between the ValidNotBefore and the ValidNotAfter.

7.11.4 DSMO alerts

Table 97 describes the DSMO alerts.

Table 97 – DSMO alerts

Standard object type name(s): Device security management object (DSMO)					
Standard object type identifier: 125					
Description of the alert: Security alerts on the state of the communication					
Alert class (Enumerated: alarm or event)	**Alert category (Enumerated: device diagnostic, comm. diagnostic, security, or process)**	**Alert type (Enumerated: based on alert category)**	**Alert priority (Enumerated: urgent, high, med, low, journal)**	**Value data type**	**Description of value included with alert**
0 = Event	2 = Security	0 = Security_DPDU_Fail_Rate_Exceeded	6 = Medium	Type: Unsigned16 Default value: 0	Alert generated after the preconfigured DPDU failure rate threshold is exceeded. The value conveys the number of failures during the time period
0 = Event	2 = Security	1 = Security_TPDU_Fail_Rate_Exceeded	6 = Medium	Type: Unsigned16 Default value: 0	Alert generated after the preconfigured TPDU failure rate threshold is exceeded. The value conveys the number of failures during the time period
0 = Event	2 = Security	2 = Security_Key_Update_Fail_Rate_Exceeded	6 = Medium	Type: Unsigned16 Default value: 0	Alert generated after the preconfigured updating security key failure rate threshold is exceeded. The value conveys the number of failures during the time period

8 Physical layer

8.1 General

The physical layer (PhL) is responsible for converting the digital data information into, and from, radio frequency energy emitted, and captured, by a device's antenna. Clause 8 also specifies the operating frequencies, transmission power levels, and modulation methods used. As described in 5.2.6.2, this standard uses IEEE 802.15.4 2,4 GHz DSSS as the default PhL, which it refers to as the Type A field medium (see 5.2.6.4). Future versions of this standard may define alternate physical layers.

The PhL provides two services, the PhL data service and the PhL management service. These services are collectively accessible via the PhSAP. The PhL data service (PhD) enables the transmission and reception of actual user data (PhPDUs) across the physical radio channel. The PhL management service is used to control the operating functions of the radio such as channel selection, transmit power selection, etc.

The structure of PhPDUs used by this standard is defined in IEEE 802.15.4. Each PhPDU consists of a PHY synchronization header (PSH), a PHY coding header (PCH), and a PHY payload that is a single PhSDU. A start frame delimiter (SFD) in the SHR is commonly used as an observable timing reference.

A device employing a certified IEEE 802.15.4-compliant 2,4 GHz DSSS radio generally will be allowed to operate without a site license in most countries around the world. Type licensing of the device that is acceptable to the country(ies) of intended use may be required.

8.2 Default physical layer

8.2.1 General requirements

The default physical layer shall be the Type A PhL (see 5.2.6.4), which shall be based on IEEE 802.15.4 2,4 GHz DSSS with additional requirements and exceptions as specified herein.

Devices on mobile platforms such as ships and trains and even trucks may move between different regulatory jurisdictions. In all cases the regulations for the current locale of the device apply. One of the device configuration parameters, dlmo.CountryCode (9.1.15.6), provides the locale and regulatory-constraint guidance needed to drive conformance to the relevant regulations; thus for mobile platforms the value of this parameter shall be changed during system operation, in a timely manner, as necessary to comply with relevant regulations. See 5.2.5.3 and Annex V.

The device vendor and the end user are responsible for certifying that these devices are compliant with this standard and any country- or region-specific regulations.

8.2.2 Additional requirements of IEEE 802.15.4

8.2.2.1 Over-the-air data rate

The PhL shall support a raw (over-the-air) data rate of 250 kbit/s.

8.2.2.2 Timing requirements

This standard requires that the PhLE support changing the channel for every PhPDU transmitted. Timing requirements are specified in Table 98.

Table 98 – Timing requirements

Event	Requirement
Time to change RF channels	< 200 μs
Time to switch from receive to transmit (with PA on)	< 200 μs
Time to switch from transmit (with PA on) to receive	< 200 μs
Inter-reception preparation time	< 200 μs
NOTE PA refers to any RF power amplifier in the apparatus.	

8.2.2.3 Carrier sense mode selection

IEEE 802.15.4 2,4 GHz DSSS physical layer supports the use of a CSMA/CA scheme to reduce collisions and increase coexistence. This scheme can delay transmission of a PhPDU excessively, due to repeated random back-off delays during channel acquisition.

The PhLE shall select the mode of CSMA/CA operation on a D-transaction by D-transaction basis as requested by the DLE, where that selection depends on the system configuration,

including the regulatory regime under which the wireless system is operating, as constrained by dlmo.CountryCode (9.1.15.6).

8.2.2.4 Number of channels

The PhLE shall support as a minimum IEEE 802.15.4 2,4 GHz DSSS channels 11..25. Support of IEEE 802.15.4 channel 26 is optional where its use is permitted by regulatory constraints, and prohibited where regulations prohibit its use.

NOTE Use of channel 26 is optional due to commonly encountered regulatory constraints near the band edge.

8.2.2.5 Transmit power limits

As specified by IEEE 802.15.4, the PhLE shall support a minimum full power level of -3 dBm, measured in accordance with the regulations to which the device is being certified.

The PhLE shall provide adjustable transmit power from -5 dBm to the maximum power of the device, in increments as specified by the PhE's TXPowerTolerance attribute, as specified in the IEEE 802.15.4 PHY PIB.

As required by IEEE 802.15.4:2011, 8.1.5, the maximum radiated power level shall not exceed the regulatory requirements that apply where the device is deployed, as constrained by dlmo.CountryCode (9.1.15.6).

8.2.3 Exceptions to the IEEE 802.15.4 physical layer

8.2.3.1 General

The requirements of this standard that are deviations or omissions from the IEEE 802.15.4 physical layer are listed here.

8.2.3.2 Limitation of frequency bands and modulation classes

Although the IEEE 802.15.4 physical layer supports multiple frequency bands and modulation classes, a device compliant with this standard shall operate in the license-exempt 2 400..2 483,5 MHz band using DSSS modulation (and coding at 250 kbit/s), which is specified in IEEE 802.15.4:2011, Table 66, as 2 450 DSSS. This standard does not support any of the other frequency bands or data rates or modulation and coding techniques specified in IEEE 802.15.4.

9 Data-link layer

9.1 General

9.1.1 Overview

The data-link layer (DL) in this standard is designed with the general goal of constraining the range of recognized construction options for a field device, while enabling flexible and innovative system solutions.

The DL specification provides a set of capabilities that are well defined and verifiable for each device that participates in a D-subnet. The DLE can be conceptualized as a table-driven state machine that operates independently on each device. A D-subnet is a group of DLEs provided with a matched set of table-driven configurations by the system manager.

The DL's building blocks include timeslots, superframes, links, and graphs. The system manager may assemble these building blocks to configure a DLE in one of three general operational alternatives, slotted-channel-hopping, slow-channel-hopping, and hybrid slotted/slow-channel-hopping. (See 9.1.7.2 for a discussion of channel-hopping.)

A timeslot is a single, non-repeating period of time. The timeslot durations in this standard are configurable to a fixed value such as 10 ms or 12 ms. Once a timeslot duration is selected, all timeslots generally have the same duration and they are re-aligned to a 4 Hz cycle at each 250 ms clock interval. (See 9.1.9 for a discussion of DLE timekeeping.)

A superframe is a collection of timeslots repeating on a cyclic schedule. The number of timeslots in a given superframe determines how frequently each timeslot repeats, thus setting a communication cycle for DLEs that use the superframe. The superframe also has an associated reference channel-hopping pattern. (See 9.1.8 for a discussion of superframes.)

Links are connections between DLEs. When the system manager defines paths between DLEs, the DLEs receive link assignments. A link assignment repeats on a cyclic schedule, through its connection to an underlying superframe. Each link refers to one timeslot or a group of timeslots within a superframe, its type (transmit and/or receive), information about the DLE's neighbor (the DLE on the other end of the link), a channel offset from the superframe's underlying channel-hopping pattern, and transmit/receive alternatives.

This standard supports graph routing as well as source routing. A directed graph is a set of directed links that is used for routing Data DPDUs within a D-subnet. Each directed graph within the D-subnet is identified by a graph ID. In source routing, the originating DLE designates the hop-by-hop route that a Data DPDU takes through a D-subnet. Graph routing and source routing may be mixed. (See 9.1.6 for a discussion of routing.)

9.1.2 Coexistence strategies in the DL

This standard incorporates several strategies that are used simultaneously to optimize coexistence with other users of the 2,4 GHz radio spectrum, as described in 4.6.10. Most of these strategies are handled adaptively by the DLE in conjunction with the system manager.

9.1.3 Allocation of digital bandwidth

The DLE is a table-driven state machine that provides prioritized access to digital bandwidth for directional communication among DLEs within a D-subnet. The state machine operates on one timeslot at a time.

Digital bandwidth is allocated by a system management function. For example, a field device may need to report every 10 s. A level of service is arranged through the system management function, to ensure that the digital bandwidth is available when needed. The system management function, in turn, arranges the digital bandwidth available to the field device's DLE and any required intermediate router DLEs.

A link is the basic unit of service within the DL. A link may be incoming, outgoing, or bidirectional. It may be unicast or broadcast (see 9.1.9.4.2).

DL digital bandwidth may be allocated to deliver an average level of service, for example, 10 Data DPDUs of available digital bandwidth (multi-hop) per minute. Alternatively, DL digital bandwidth may be allocated to support a reporting interval and a level of service, for example, one Data DPDU (including retries) every 15 s, with 2 s maximum latency to a backbone connection.

DL digital bandwidth may be organized as a pool that can be shared by a collection of DLEs using the corresponding links. A level of service may be delivered by ensuring that sufficient shared, contention-based capacity is available.

For more granular channel allocation, links may be tied to particular groups of Data DPDUs.

A service level may be delivered with a combination of specific link allocations and generally available shared digital bandwidth. For example, for each report, a dedicated link may be

allocated for the first transmission to each of two neighbors, with retries using a shared digital bandwidth.

9.1.4 Structure of the DPDU

The general structure of a data-link protocol data unit (DPDU) in this standard is shown in Figure 53.

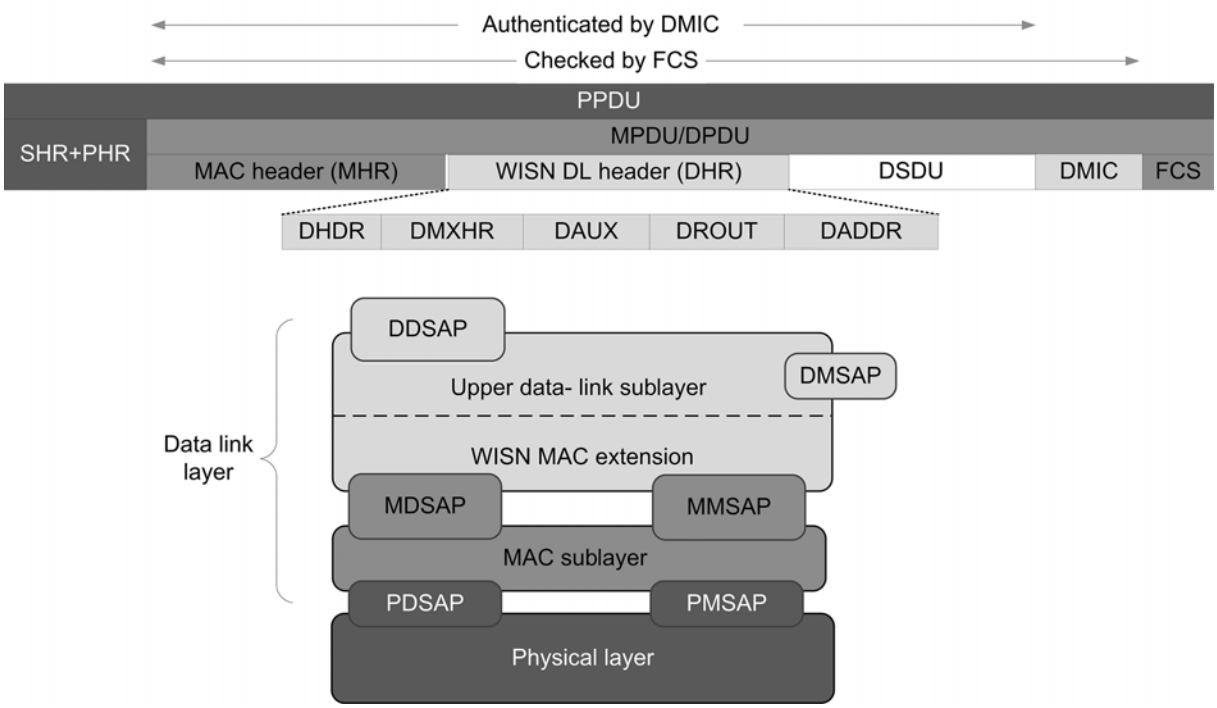


Figure 53 – DL protocol suite and PhPDU/DPDU structure

The DL specified by this standard includes:

- A subset of the IEEE 802.15.4 MAC, as described in 9.1.5. This handles the low-level mechanics of sending and receiving individual DPDUs (all of which are classified as “data” DPDUs by IEEE 802.15.4). The SHR, PHR, MHR, and frame check sequence (FCS) of every DPDU are as described and specified in IEEE 802.15.4.
- An extension to the MAC, including aspects of the DL that are not specified by IEEE but are logically MAC functions.
- An upper-DL protocol that handles link and mesh aspects above the MAC level.

Components of the DPDU header in this standard are described in 9.3.1.

9.1.5 The DL and the IEEE 802.15.4 MAC

This standard uses the IEEE 802.15.4 MAC (called IEEE MAC herein). Only IEEE MAC data frames are used. The formats used are as specified by IEEE 802.15.4, with the two exceptions explicitly enumerated in 9.1.5. See 9.3.3 for detail.

The IEEE MAC describes various features that are not used by this standard’s DL (called “the DL” herein). In summary, only IEEE MAC data frames are used by the DL.

A DLE compliant with this standard never associates with a coordinator in the sense defined by the IEEE MAC. None of the IEEE MAC functions involving FFDs are used by this standard.

Within the limited context of this standard's DPDUs, there are some features that are not supported for IEEE MAC data frames. These features are implemented via the MAC extension of this standard, which enhances the IEEE MAC with features that are logically MAC functions but that are not included in IEEE 802.15.4.

The DL and the IEEE 802.15.4 MAC each specify an entity called a superframe (see 9.1.8), but the DL uses no aspects of the IEEE 802.15.4 MAC superframe specification.

ACK/NAK DPDUs are used to convey time information for clock correction, in addition to providing authenticated acknowledgment. These features are not available when using the IEEE 802.15.4: MAC immediate acknowledgment MPDU. For this and other reasons, the MAC-level immediate acknowledgments specified in IEEE 802.15.4 are not used; instead, MAC-level immediate acknowledgments compliant with this standard are provided within this standard as short IEEE 802.15.4 data frames, usually using an address field structure combination not used for such purposes in IEEE 802.15.4.

NOTE ACK/NAK DPDUs are short control signaling (SCS) as specified in ETSI EN 300 328.

The IEEE 802.15.4 MAC includes active and passive scans, which are not used in this standard. This standard has alternative active and passive scans, using IEEE 802.15.4 data frames.

The IEEE 802.15.4 MAC backoff and retry mechanism is not used by the DL. Instead, the DL implements its own retries, involving spatial diversity (retries to multiple DLEs), frequency diversity (retries on multiple radio channels), and time diversity (delaying the Data DPDU). The manner and degree of these elements of diversity are not fixed, but configured by the system manager. More generally, this standard's DL uses CSMA/CA, but the details are different from CSMA/CA use as defined in IEEE 802.15.4. Various aspects of the IEEE 802.15.4 MAC's CSMA/CA behavior are not used, and CSMA/CA functions are handled in this standard's DL.

The standard includes two exceptions to the MAC-PDU addressing combinations specified in IEEE 802.15.4:

- Solicitation Data DPDUs and most ACK/NAK DPDUs, which are technically data frames in IEEE 802.15.4, use a destination-addressing mode of 00 and a source-addressing mode of 00. In IEEE 802.15.4, this combination is limited to IEEE 802.15.4 beacons and trivially-spoofed IEEE 802.15.4 immediate acknowledgments.
- Advertisement Data DPDUs and secondary duocast/N-cast ACK/NAK DPDUs, which are technically data frames in IEEE 802.15.4, use a destination-addressing mode of 00 and a source-addressing mode of 10 (DL16Address). In IEEE 802.15.4, this combination implies that the frame is directed to the PAN coordinator, which does not exist in this standard (so therefore that meaning cannot apply).

9.1.6 Routes and graphs

9.1.6.1 General

Routes are configured by the system manager, based on reports from DLEs that indicate instantaneous and historical quality of wireless connectivity to their immediate neighbors. The system manager accumulates these reports of signal quality to make routing decisions. The signal quality reports are standardized, but the routing decision process within the system manager is not standardized. Once the system manager makes its routing decisions, it uses standard Data DPDUs to configure routes within each DLE in the D-subnet. (See 9.1.13 and 9.1.14 for a review of neighbor discovery.)

DL routing is adaptive at two levels:

- DLEs make instantaneous adaptive forwarding decisions. DLEs are normally configured with path diversity, so that if one link fails somewhere along the route, the DLE can immediately send the Data DPDU along an alternative path.
- If, over time, certain links have consistent connectivity issues, this is reported to the system manager, which can then reconfigure the DLE to use different links.

Within each Data DPDU, DL routing instructions are placed in the Data DPDU's DROUT subheader (see 9.3.3.6). When a Data DPDU is addressed to an immediate neighbor, such as during the D-subnet joining process, the route is simply the address of that neighbor. When the Data DPDU is being sent to a more distant DLE, a single graph number can indicate how the Data DPDU's payload should be conveyed through the D-subnet to reach that address. These two approaches may be combined. For example, a DROUT subheader may contain two entries, the first identifying an immediate neighbor for the first hop, and the second indicating a graph that is used for the rest of the route through the D-subnet.

Routes that identify specific D-addresses, also known as source routing, are not as adaptive as routes based on graphs. When a route is based on a series of D-addresses, each DLE along the route becomes a single point of failure. Graphs, on the other hand, should be configured with multiple branches at each hop, so that if there is a connectivity problem with one neighbor, the DLE can send the payload of that Data DPDU to a different neighbor.

Source routing is useful for quick, transitory communications between DLEs, such as during the joining process. Source routing may also be used when graph route resources are scarce.

The DROUT subheader is constructed by the DLE that injects the Data DPDU into the DL from a NLE. Routes are selected by table lookup based on contract ID, destination D-address, or by default. Once a route is selected the Data DPDU's conveyed payload follows that route until it arrives at the D-subnet termination point, which may be its ultimate destination, or alternatively may be a waypoint along the route, such as a backbone router or the system manager. At the D-subnet termination point, the received Data DPDU's payload is passed to the collocated NLE.

The DROUT subheader includes a forwarding-limit field which is used to limit the number of times that a Data DPDU can be forwarded within a D-subnet. The forwarding limit is initialized by the originating DLE when the route is assigned, and decremented with each hop until it reaches zero, triggering discard of the Data DPDU if a subsequent hop was required.

NOTE This forwarding limit ensures that Data DPDUs cannot circulate forever via an unintended circular route.

9.1.6.2 Graph routing

A graph is a set of directed links that is used for routing messages within a D-subnet. Each graph designated by the system manager for routing within a D-subnet is identified by a graph ID.

The links associated with each graph are configured by the system manager. A D-subnet may have multiple graphs, some of which may overlap. Each DLE may have multiple graphs going through it, even to the same neighbors.

Figure 54 illustrates an example of graph routing.

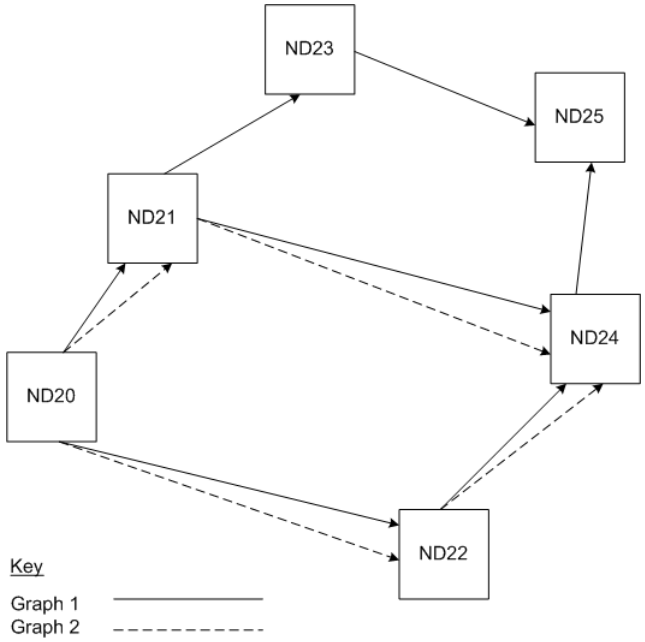


Figure 54 – Graph routing example

In Figure 54, ND20 communicates with ND25 using graph 1. To send a Data DPDU's payload on that graph, ND20 may forward it to ND21 or ND22. From those DLEs, the Data DPDU's payload may take several alternate routes, but either way, following graph 1, the Data DPDU's payload will arrive at ND25. Similarly, to communicate with ND24, ND20 may send Data DPDUs on graph 2 through ND21 or ND22, either of which in turn will forward the Data DPDU's payload to ND24.

Figure 54 shows all graphs originating from ND20, but the same graphs may be used by any node. For example, the system manager may configure ND21 to use graph 1 for its communication with ND25.

NOTE 1 The DL routing information in the Data DPDU's payload often is updated as that payload moves through the D-subnet.

Table 99 and Table 100 reflect the contents of graph tables on ND20 and ND21. These graph tables roughly correspond to data structures within each DLE for the topology shown in Figure 31. For example, a Data DPDU following graph 2 will look up graph ID 2 in each router along the route to find out which neighbors it can use for the next hop.

Table 99 – Graph table on ND20

Graph ID	Neighbor address
1	21, 22
2	21, 22

Table 100 – Graph table on ND21

Graph ID	Neighbor address
1	23, 24
2	24

Each graph within a D-subnet is identified by a graph ID. A Data DPDU usually originates within the D-subnet, at a field device or at a gateway or system manager or backbone router.

To send a message on a graph, the originating DLE includes a graph ID in the Data DPDU's DROUT subheader. The Data DPDU travels along the paths corresponding to the graph ID until it reaches its destination or is discarded.

In order to route Data DPDUs over a graph, each DLE along the path needs to maintain a graph table containing entries that include the graph ID and the next-neighbor(s)' D-address(es). A DLE routing a Data DPDU performs a lookup based on graph ID and sends the Data DPDU to any one of the applicable neighbors. Once a neighbor acknowledges receipt of the Data DPDU, the DLE releases it from a Data DPDU forwarding buffer.

Diverse graph paths (branches) may be established by configuring more than one neighbor associated with the same graph index. A branch may be configured with a preferred neighbor, indicating that the DLE should attempt to transmit the Data DPDU to the preferred neighbor first, even if there is an earlier-occurring opportunity to transmit to other neighbors. If no preferred neighbor is designated, the DLE should treat all branches equally, transmitting the Data DPDU at the first opportunity that presents itself. If the first transmission does not result in an ACK DPDU, the DLE normally is configured to use alternative branches for retries.

Figure 55 provides examples of routing graphs that are inbound (toward the backbone) and outbound (away from the backbone). The basic organization of inbound and outbound routing graphs may be very similar to each other, but pointing in opposite directions, as shown in Figure 55. The system manager configures routing relationships among DLEs in a D-subnet.

The top half of Figure 55 shows an inbound graph in an example of DL routing configuration. An inbound graph enables a set of DLEs to send Data DPDUs toward the backbone or system manager. A single graph may be used for inbound routing in a small D-subnet, as shown in the top half of Figure 55. It is possible and often desirable for the system manager to define multiple inbound graphs, particularly as the number of DLEs in the D-subnet increases. As shown in the top half of Figure 55, if DLEs have multiple neighboring routers, then diversity is often inherent in the inbound graph.

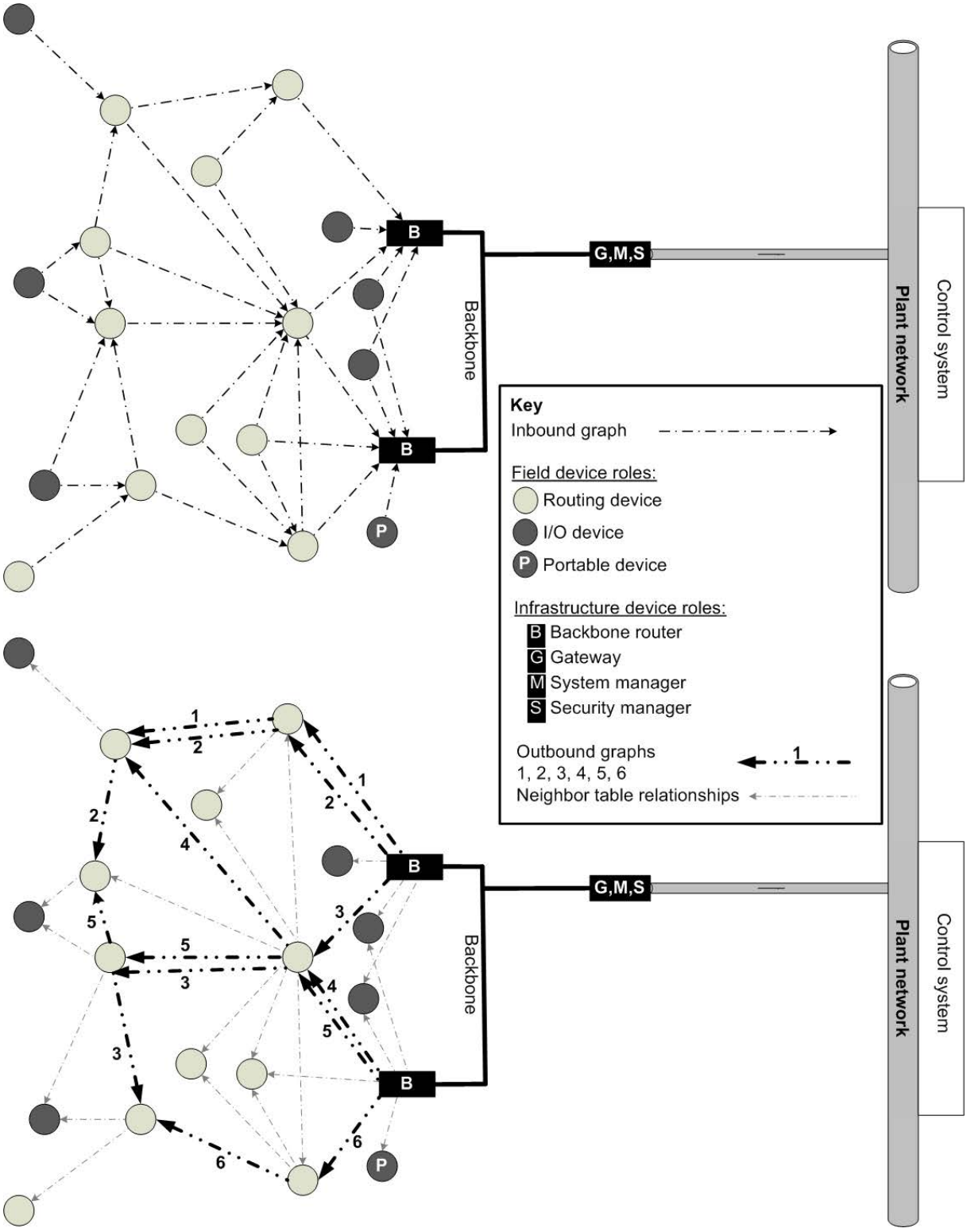


Figure 55 – Inbound and outbound graphs

The illustrative inbound graph in Figure 55 can be problematic for fragmented NPDUs, because fragments arriving at different backbone routers may pose a reassembly challenge. The following considerations apply:

a) Each contract specifies a maximum NPDU size. Most contracts covering communication toward the backbone, including those used for generally higher priority traffic such as publish/subscribe communication of process variables and source/sink communication of alerts (i.e., process alarms and device events), specify maximum APDU sizes that will not

trigger NPDU fragmentation, which thus does not occur. Most communications fit in a single Data DPDU, which is determinable for a particular flow in terms of the maximum supported payload size that is agreed when the contract is established.

b) Those contracts that permit NPDUs of a size that can require fragmentation toward the backbone are typically low-priority background transfers of large blocks of information, such as captured waveform upload. A problem occurs when the fragments of a fragmented NPDU are delivered to different BBRs.

That problem (of uncoordinated fragment delivery) can be avoided in two different ways:

1) The contract may specify or be tied to a route that terminates in a single BBR, thus avoiding delivery of different fragments of a single NPDU to differing devices.

2) The contract may specify or be tied to a route that terminates in multiple BBRs from the same vendor, that are known by the system manager to have an inter-BBR fragment reassembly protocol to manage such reassembly via their shared backbone network.

NOTE 2 It is common practice for plant owner-operators to purchase identical-function infrastructure equipment from only one vendor, so that problems with that class of equipment will be referable directly to the responsible vendor, thus reducing the need of plant personnel to determine which specific vendor's equipment is at fault. Such elimination of inter-vendor "finger pointing" typically leads to faster problem resolution and a quicker return to normal plant productivity. It also makes more usable any vendor-proprietary features or diagnostics that the infrastructure equipment may provide.

For DLE contracts supporting payload sizes that do not fit in a single DSDU, the selected graphs that are directed toward the backbone often use reduced path diversity to ensure that a set of fragmented NPDUs are all delivered to the same BBR (alternative a)) or to a subset of BBRs that are known to jointly provide a shared fragment-reassembly capability (alternative b)).

The bottom half of Figure 55 shows a set of outbound graphs in an example of routing configuration. An outbound graph is usually used to send Data DPDUs from backbone DLEs to field DLEs. As shown in the bottom half of Figure 55, multiple graphs may be used for outbound routing, with each outbound graph corresponding to a group of DLEs within radio range of the graph.

In this example, the inbound DL graph routing ends at the backbone, at which point the NL takes over routing responsibilities. The relationship between a WISN D-subnet and a backbone N-subnet is described in 5.5.

Although all of the examples above show inbound and outbound graphs, these are not actually different types of graphs; they are just graphs that happen to point in opposite directions. Peer-to-peer routing is also supported by this standard. The system manager may arrange a graph to follow any route where connectivity exists.

A DSDU is forwarded along a graph until the graph is terminated. If the Data DPDU's destination address matches the DLE's address, then the DSDU has reached its destination and the graph is terminated. Alternatively, if the graph number in the Data DPDU does not have a corresponding entry in the lookup table dlmo.Graph (see 9.4.3.6), the graph has reached its termination point.

9.1.6.3 Graph extensions

The bottom half of Figure 55 shows that outbound graphs do not necessarily extend to all DLEs on the periphery of a D-subnet. Nonetheless, such DLEs are covered by the outbound graphs implicitly, through a graph extension mechanism. A DLE automatically extends graphs by checking the Data DPDU's destination address for a neighbor table entry, thus indicating that the Data DPDU's destination is one hop away. If it is, the router treats the neighbor as if it were listed in the graph, thereby extending the graph for that Data DPDU. More formally, when the Data DPDU's destination address is in a DLE's neighbor table, and the Data DPDU is being routed with a graph, the DLE shall treat that graph as including that neighbor for the purpose of routing that Data DPDU even if the graph does not explicitly refer to the neighbor. All routers shall support this basic form of implicit graph extensions.

An explicit graph extension field in the neighbor table provides an additional degree of control. If a graph is specifically designated in a neighbor table, as described in 9.4.3.4.2, the neighbor is not only treated as being covered by the graph; the neighbor is also given preferential treatment. If the neighbor is designated as the graph's last hop, a Data DPDU following that graph shall be forwarded exclusively to that neighbor. If the neighbor is designated as a preferred branch, the DLE should attempt to forward an applicable Data DPDU to that neighbor before other neighbors.

Support for the explicit graph extension field in the neighbor table is a device construction option; its support status is reported to the system manager through dlmo.DeviceCapability when the DLE joins the D-subnet. All routers support the basic implicit graph extension capability, but it is expected that only some routers will fully support the explicit-last-hop and preferred-branch indicators in the neighbor table.

9.1.6.4 Source routing

Source routing is a general method of routing supported by this standard. In source routing, the originating DLE may be configured to designate a hop-by-hop route for a Data DPDU to follow through a D-subnet. A simple use of source routing is a Data DPDU directed one hop away to a specific neighbor, such as for joining. When a source-routed Data DPDU arrives at an intermediary DLE, the intermediary DLE examines the path information in the Data DPDU to determine the neighbor to which it should forward the Data DPDU.

A source route is a list of entries specifying the route that a Data DPDU shall follow through the D-subnet. The first entry in the list specifies the next hop, and the list is shortened as the Data DPDU moves through the D-subnet. Source routing entries can specify graphs or D-addresses, thus allowing graphs to be chained. 12-bit graph numbers within a source route are encoded in binary as 0x1010 gggg gggg gggg.

The Data DPDU header compresses a source route to a single octet in the common case where a single graph route is specified and the graph number is ≤ 255 (encoded in binary as 0x1010 0000 gggg gggg). This is commonly referred to as graph routing, but formally it is a source route containing a single graph.

In the provisioning or joining process, an EUI64Address is used to address a DLE that has not yet received an IPv6Address. This case is encoded as a route with a graph of zero and a DADDR D-address of zero (see 9.3.3.6 and 9.3.3.7), indicating that the EUI64Address can be found in the Data DPDU's MAC header (MHR).

When a Data DPDU is received by the DLE through its wireless link, the following processing steps shall be followed, in order to determine whether the DL route has terminated and to update the source route in the Data DPDU header:

- The DADDR subheaders' destination D-address is checked to see if it matches the D-address of the receiving DLE. If there is a match, the DSDU has reached its final destination and the DSDU shall be passed to the collocated NLE as described in 9.2.4. (The DADDR destination D-address is encoded as zero, in the case where the destination D-address is duplicated in the MHR. See 9.3.3.7. In that case, the match is indirect, based on the MHR destination D-address.)
- The first entry in the source route is deleted if appropriate, thus shortening the source route by shifting the second and subsequent entries (if any) into the prior positions (shift left). The first entry shall be deleted unless it is a graph number of a graph that has not reached its termination point.
- If the route has no remaining entries, the route has terminated and the DSDU shall be passed to the collocated NLE as described in 9.2.4.

If the DSDU is not passed to the collocated NLE, the Data DPDU shall be discarded if the forwarding limit (in the DROUT subheader) is zero. If the forwarding limit is positive, it is decremented and the Data DPDU placed on the DLE's forwarding message queue.

When a DSDU is intended to be routed through the backbone, the DL route should terminate at the backbone router. If a route does not terminate in a collocated backbone router, it is forwarded by the DLE and never processed by the collocated backbone router's NLE, thus allowing peer-to-peer messaging to occur within the D-subnet through the auspices of a backbone router.

These routing methods are examples of different ways to configure the DL routing capability that is resident in all field routers compliant with this standard. The system manager shall configure all graphs within a D-subnet. The ability to configure routing in any of several ways such as graph routing and/or source routing enables device interconnectability.

9.1.6.5 Route selection

The route through the D-subnet for a Data DPDU is selected when the message enters the DL (see 9.2.2). The route is stored in the DROUT subheader for use by other DLEs that will route the Data DPDU. The initial selection of a route is based on decision rules in the initiating DLE. The following list shows the route selection criteria in order, whereby the route shall be selected based on the first condition that applies:

- The Data DPDU has a destination EUI64Address. A destination EUI64Address is used only during the joining process, when a router is sending a response to an immediate neighbor that has not yet received a DL16Address from the system manager. In that case, the EUI64Address from the IEEE MAC is used, with a Graph ID of 0 in the D-route.
- The ContractID is associated with a particular D-route. This may be used when a particular graph or source D-route is intended to provide a defined level of service.
- The destination DL16Address within the D-subnet is associated with a particular route.
- The destination DL16Address within the D-subnet is an immediate neighbor, such as during the joining process.
- Otherwise, use the default route. Normally, the default will direct the message to the nearest backbone router, or to the system manager if there is no backbone router.

A single route may be designated as the default by the system manager, by designating a particular D-route as a default. The default D-route is usually configured to route messages to the system manager, or to a backbone router if there is no system manager on the D-subnet. A default D-route may sensibly be configured in conjunction with the establishment of a DLE's contract with the system manager. Additional routes may be configured as needed, such as to provide enhanced quality of service or to route messages to a peer DLE on the D-subnet.

9.1.7 Slotted-channel-hopping, slow-channel-hopping, and timeslots

9.1.7.1 General

Three general operational alternatives are supported by the DL:

– slotted-channel-hopping;

– slow-channel-hopping; and

– hybrid combinations of slotted-channel-hopping and slow-channel-hopping.

These three operational alternatives are different ways for a system manager to configure a slotted-channel-hopping capability that is supported by every DLE in a D-subnet. Channel-hopping schedules are configured by the system manager through advertisement Data DPDUs and the dlmo.Superframe attribute.

Slotted-channel-hopping and slow-channel-hopping provide different ways to configure a series of timeslots. The system manager determines the mode of operation and assigns the use of superframes, which are cyclic collections of timeslots. (See 9.1.8 for further discussion

of superframes.) This provides flexibility and interoperability of the relevant communication functions (i.e., interconnectability) without requiring excessive complexity within the devices.

From the perspective of a field device, the DLE can be visualized as a player piano with several keys. Each key corresponds to a D-transaction, which specifies a specific channel and timeslot. One key is used to send a pending Data DPDU from the outbound queue, another key is used to listen for an incoming Data DPDU, and so forth. The system manager provides a piano roll for the DLE to play over and over again. The playing style may be slow-channel-hopping, slotted-channel-hopping, or a hybrid of the two. The DLE does not differentiate; it simply mechanically plays each key at a specified time on a specified channel, based on the instructions on the piano roll.

The note details within a timeslot are configurable, using timeslot templates provided by the system manager. There is a constrained series of operations that can be performed within a timeslot – transmit, listen, wait, timeout, and acknowledge – but those simple building blocks may be assembled with different timings. These definitions are flexible, under the control of the system manager.

The duration of timeslots in a D-subnet is set to a specific value by the system manager when a DLE joins the D-subnet. A timeslot duration of 10 ms to 12 ms is expected to be typical. Timeslot duration is configurable to enable:

- optimized coexistence with other systems, such as other D-subnets conforming to this standard, to IEC 62591, and to IEC 62601;
- longer timeslots to accommodate extended message wait times;
- shorter timeslots to take full advantage of optimized implementations;
- longer timeslots to accommodate serial ACK/NAK DPDUs from multiple devices (e.g., duocast, N-cast);
- longer timeslots to accommodate long-duration CSMA/CA at the start of a timeslot (e.g., for prioritized access to shared timeslots);
- longer timeslots to accommodate slow-hopping periods of extended duration;
- timeslots to be synchronized with other non-standard-compliant D-subnets to facilitate inter-routing.

Figure 56 illustrates a slotted-channel-hopping operation.

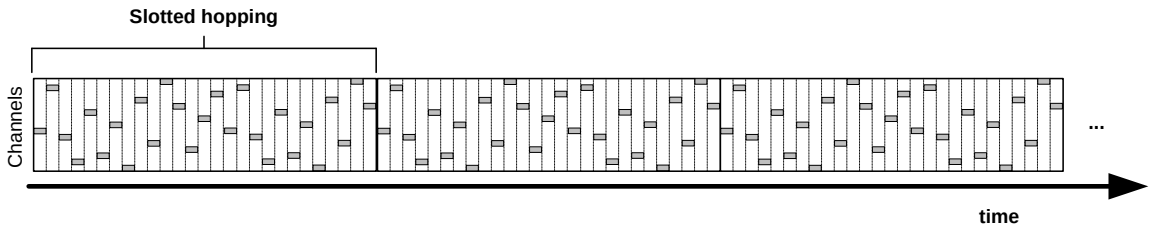


Figure 56 – Slotted-channel-hopping

In slotted-channel-hopping, channel-hopping timeslots of equal duration are used. Each timeslot uses a different radio channel in a channel-hopping pattern. In slotted-channel-hopping, each timeslot is intended to accommodate a single D-transaction consisting of one Data DPDU and its ACK/NAK DPDU acknowledgment(s).

Figure 57 illustrates a slow-channel-hopping operation.

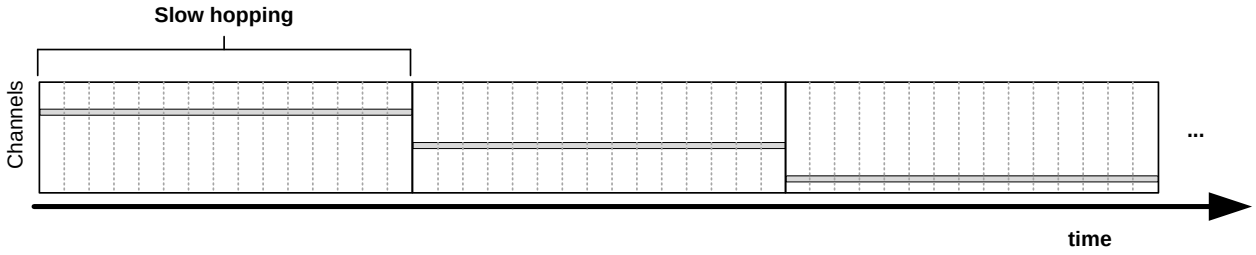


Figure 57 – Slow-channel-hopping

In slow-channel-hopping, a collection of contiguous timeslots is grouped on a single real radio channel. Each collection of timeslots is treated as a single slow-channel-hopping period; however, as shown in Figure 57, timeslots still underlie slow-channel-hopping. Slow-channel-hopping periods are configurable, usually on a scale of about 100 ms to 400 ms per hop. Longer slow-channel-hopping periods, potentially multiple seconds in duration, may be used to support DLEs with imprecise timekeeping and/or DLEs that have temporarily lost contact with the D-subnet. To enable DLE interworkability, all DLEs compliant with this standard shall support configuration of timeslot duration, as designated by the system manager.

In some regulatory domains, slow-channel-hopping of an IEEE 802.15.4 2,4 GHz radio is not permitted to exceed 400 ms per hop. However, in other regulatory domains there is no such constraint. The slow-channel-hopping period is set by the system manager, not by this standard, but its use is constrained in each DLE by dlmo.CountryCode (9.1.15.6). Thus this regulatory constraint is explicitly enforced where it exists, similar to other regulatory constraints.

The structure, duration, and assignment of timeslots in slow-channel-hopping periods are configured by the system manager, which determines the mode of operation and assigns the use of timeslots. This provides flexibility without requiring excessive complexity within the DLEs, facilitating DLE interworkability.

Figure 58 illustrates a hybrid mode of operation.

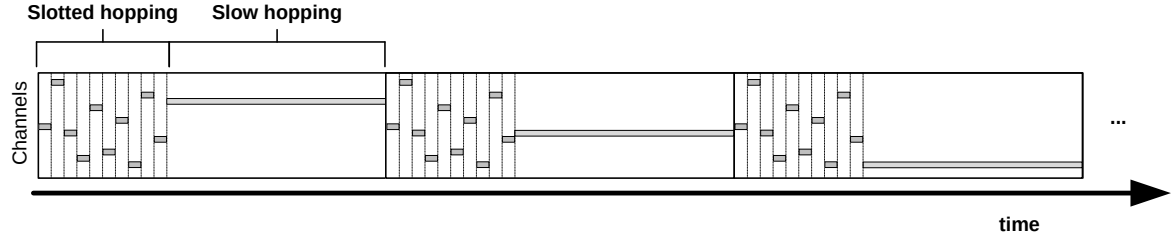


Figure 58 – Hybrid operation

Hybrid operation uses slotted-channel-hopping and slow-channel-hopping periods in a configured combination. For example, in Figure 58, a number of timeslots using slotted-channel-hopping are followed by a period of slow-channel-hopping.

9.1.7.2 Channel-hopping

9.1.7.2.1 General

DL communications are intended to be distributed across multiple radio channels. The system uses defined channel-hopping patterns which provide a specific sequence of channels for communication among collections of devices. Channel-hopping begins at a designated offset in the channel-hopping pattern, continuing through the pattern sequentially until the end, then repeating the pattern indefinitely.

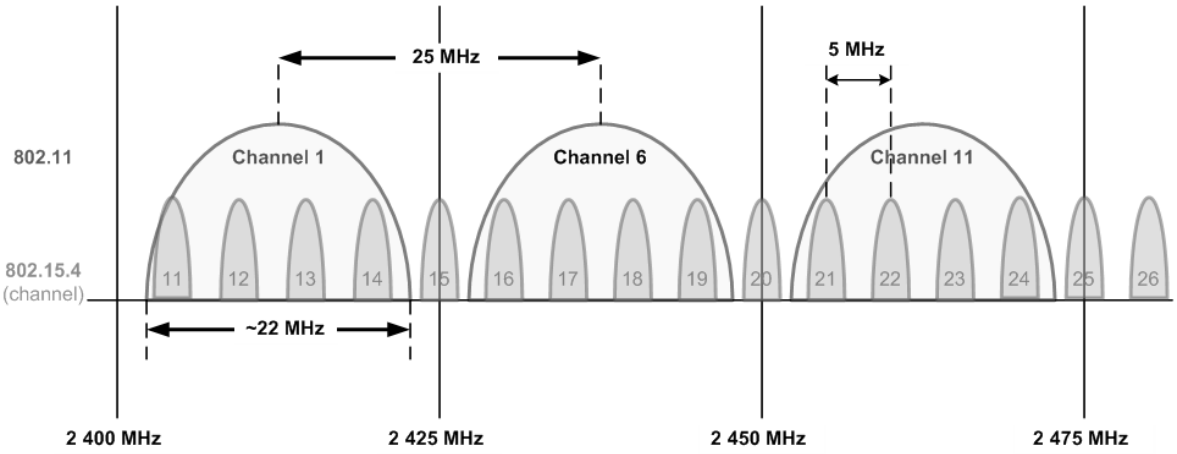
IEEE 802.15.4 DSSS channels 11..26 are mapped to nominal channel numbers 0..15 in this standard. This overview refers to them by their IEEE 802.15.4 nomenclature, as channels 11..26.

This standard is based on devices that use one channel at a time. Multiple radios that are co-packaged are able to operate multiple instances of the PhLE and DLE simultaneously on different channels. The details of such multi-channel operation are not specified by this standard, but such operation is intentionally supported in the D-nonce.

9.1.7.2.2 Radio spectrum considerations

For radio communication, this standard uses IEEE 802.15.4 DSSS channels in the 2,4 GHz band. The IEEE 802.15.4 physical layer (2,4 GHz, DSSS) includes sixteen channels, numbered 11 through 26.

Figure 59 illustrates the sixteen IEEE 802.15.4 DSSS channels along with three overlapping, commonly used IEEE 802.11 channels.



NOTE Channel numbers shown are those of IEEE 802.11 and IEEE 802.15.4, rather than those of this standard.

Figure 59 – Radio spectrum usage

In Figure 59, the narrow channels 11..26 are IEEE 802.15.4 2,4 GHz DSSS channels; these channels are substantially non-overlapping. Also shown are the wider IEEE 802.11 channels 1, 6, and 11; these three channels are common choices for IEEE 802.11 communication and each overlaps a number of IEEE 802.15.4 2,4 GHz DSSS channels.

As Figure 59 shows, IEEE 802.15.4 2,4 GHz DSSS channels 15, 20, and 25 do not substantially overlap any of the three common IEEE 802.11 channels. Therefore, IEEE 802.15.4 2,4 GHz DSSS channels 15, 20, and 25 may reasonably be designated as slow-channel-hopping channels. These slow-channel-hopping channels may be configured for purposes such as neighbor discovery. For example, information about the D-subnet may be advertised by field routers on pre-designated slow-channel-hopping channels, so that any DLE seeking to join the D-subnet can limit its active and passive scans to these channels when discovering nearby field routers.

This illustration demonstrates how spectrum management techniques supported by this standard may be used to account for a commonly encountered scenario. Alternative configurations of WiFi, other users of the radio spectrum, and local regulatory restrictions may justify alternative configurations in actual installations.

Most DL communications are intended to be distributed across all available IEEE 802.15.4 DSSS channels (11..26). This standard defines a number of predefined channel-hopping

patterns, each providing a specific sequence of channels to use. Other channel-hopping patterns are configurable by the system manager. The channel-hopping patterns that are predefined in this standard are selected to have certain properties intended to minimize the occurrence of unmanaged and repeated collisions with co-located wireless devices, particularly IEEE 802.11 devices.

9.1.7.2.3 Channel 26 and other blocked channels

Support for IEEE 802.15.4 2,4 GHz DSSS channel 26 is optional in this standard, because some implementations may encounter regulatory restrictions at the upper band edge. In addition, a DLE may be blocked from using other channels due to regulatory restrictions, but not for other reasons. After restriction or forced-enabling for the regulatory domain specified by dlmo.CountryCode (9.1.15.6), a list of channels that the DLE supports is reported to the system manager during the process of the DLE joining the D-subnet. This is done through the attribute dlmo.DeviceCapability.ChannelMap. A DLE may be configured with links that use such unsupported channels; in that case the DLE shall treat those links as unselectable.

Since support for IEEE 802.15.4 2,4 GHz DSSS channel 26 is optional in the standard, a system manager may sensibly limit D-subnet operation to IEEE 802.15.4 2,4 GHz DSSS channels 11..25. The channel-hopping patterns predefined by this standard include IEEE 802.15.4 2,4 GHz DSSS channel 26, but these predefined channel-hopping sequences are designed so that they can be shortened by excluding IEEE 802.15.4 2,4 GHz DSSS channel 26 from the channel map in each superframe.

9.1.7.2.4 Spectrum management and selective channel utilization

Multiple methods are available for limiting use of busy or undesirable radio channels, including clear channel assessment (CCA), spectrum management, and selective channel utilization.

Timeslots are normally configured to check for a clear channel before transmitting, using the different modes of the CCA mechanism defined in IEEE 802.15.4. CCA causes a DLE that is about to initiate transmission to relinquish a timeslot if use of the channel by another DLE is detected prior to transmission. See 4.6.11, 9.1.9.4.3 and 9.1.9.4.8.

Spectrum management is a form of selective channel utilization. Spectrum management limits the DL configuration to a subset of channels. Limiting slow-channel-hopping to IEEE 802.15.4 2,4 GHz DSSS channels 15, 20 and 25 is an example of spectrum management. Another example is when a system manager blocks (blacklists) certain radio channels that are not working well or are prohibited by regulation or local policy, or whitelists channels that are mandated by regulation or local policy. Spectrum management is handled by the system manager, through the way that it configures a DLE and the associated PhLE. See 9.1.8.4.7.

Additionally, a DLE may autonomously treat transmit links on problematic channels as idle, thus reducing unnecessary interference and wasted energy on channels with a history of poor connectivity. A DLE skipping links in this manner should periodically test the links to verify that they remain problematic. Such selective channel utilization can be disabled by the system manager on a link-by-link basis, through the attribute dlmo.Link[ ].Type.SelectiveAllowed. See 9.4.3.7.2, Table 182.

9.1.7.2.5 Repeating channel-hopping-patterns

This standard supports five predefined IEEE 802.15.4 2,4 GHz DSSS repeating channel-hopping patterns, which shall be supported in every DLE:

- pattern1: 19, 12, 20, 24, 16, 23, 18, 25, 14, 21, 11, 15, 22, 17, 13 (, 26);
- pattern2: pattern1 in reverse order;
- pattern3: 15, 20, 25 (intended for slow-channel-hopping channels);

- pattern4: 25, 20, 15 (pattern3 in reverse order);
- pattern5: 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21. 22, 23, 24, 25 (, 26).

NOTE 1 IEEE 802.15.4 2,4 GHz DSSS channel 26 is shown in parentheses, as it is supported by this standard but is not necessarily used due to commonly encountered regulatory constraints at the band edge. Figure 60 through Figure 66 and Figure 70 through Figure 74 mostly include channel 26, even though its use is commonly masked out by the superframe that uses the hop sequence.

NOTE 2 In this standard channels are numbered 0..15, as described in 9.4.3.2. However, for tutorial purposes and to ease comparison with IEEE 802.11 (WiFi) and other uses of the same frequency band, channel-hopping patterns are expressed as their IEEE 802.15.4 DSSS channel numbers.

NOTE 3 Pattern5, which is based on IEC 62591, is intended to facilitate coexistence with that IEC standard.

The system manager can configure a DLE to use any of these channel-hopping patterns for slotted-channel-hopping, slow-channel-hopping, or hybrid channel-hopping.

Any channel or set of channels in a channel-hopping pattern may be disabled (masked out) by configuration of the superframe that uses the channel-hopping pattern, with the effect of shortening the channel-hopping pattern for spectrum management.

The predefined channel-hopping patterns of this standard are designed to have certain properties, whether channel 26 is included in the pattern or not. Specifically, successive channels in most of the predefined channel-hopping-patterns are separated by at least 15 MHz, the same bandwidth as three IEEE 802.15.4, 2,4 GHz DSSS channels. This property mitigates the effects of interference and multipath fading in industrial environments.

As shown in the example in Figure 60, predefined channel-hopping-pattern1 is arranged so that consecutive hops do not overlap the same IEEE 802.11 channel.

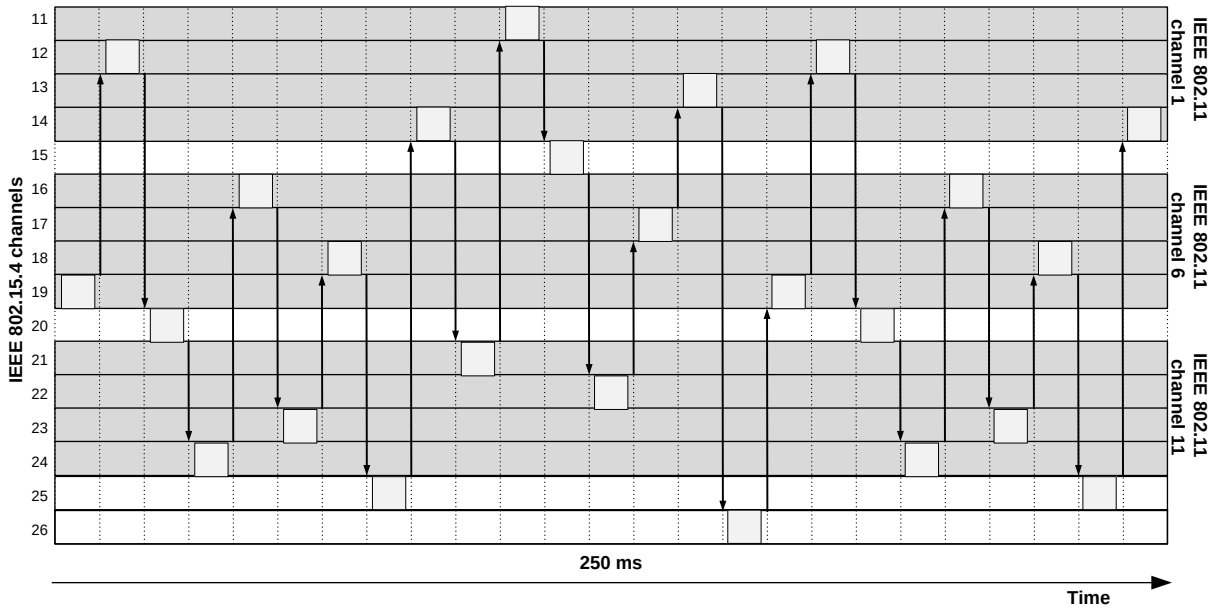


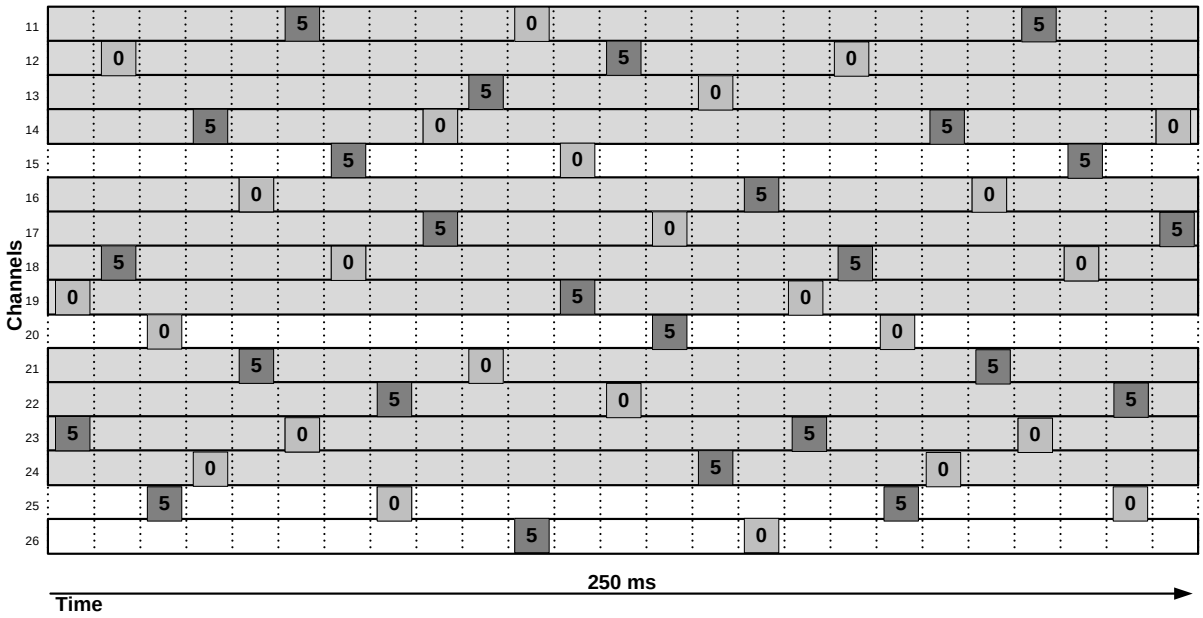
Figure 60 – Predefined channel-hopping-pattern1

At least three channels separate each consecutive hop in pattern1, resulting in frequency shifts of at least 20 MHz. When retries occur in consecutive hops, they will not encounter or cause interference in the same IEEE 802.11 channel.

For different groups of DLEs, it is desirable for DLEs to hop on non-interfering patterns.

Each channel-hopping pattern is combined with a hopping pattern offset. If the hopping pattern offset is zero, then the specified baseline channel-hopping pattern is used. If the

hopping-pattern offset is 5, then an offset of 5 is used when indexing the baseline channel-hopping pattern. Figure 61 shows how two groups of DLEs with different hopping-pattern offsets into channel-hopping pattern1 may be used together without competing for the same radio channel at the same time.



NOTE Channel numbers shown are those of IEEE 802.15.4, rather than those of this standard.

Figure 61 – Two groups of DLEs with different channel-hopping-pattern-offsets

A superframe's channel-hopping-offset is determined indirectly from the dlmo.Superframe[ ].ChBirth attribute, called simply ChBirth, which gives the starting reference of the channel-hopping pattern at TAI time zero. This provides a baseline channel-hopping sequence. Offsets from a baseline channel-hopping sequence, given in the dlmo.Link[ ].ChOffset attribute, called simply ChOffset, are as described here and shown in Figure 61. While the same result is achievable by using ChBirth or ChOffset, it should be noted that the two attributes are essentially reversed. Details of channel-hopping-pattern-offset calculations are found in 9.4.3.5.3.

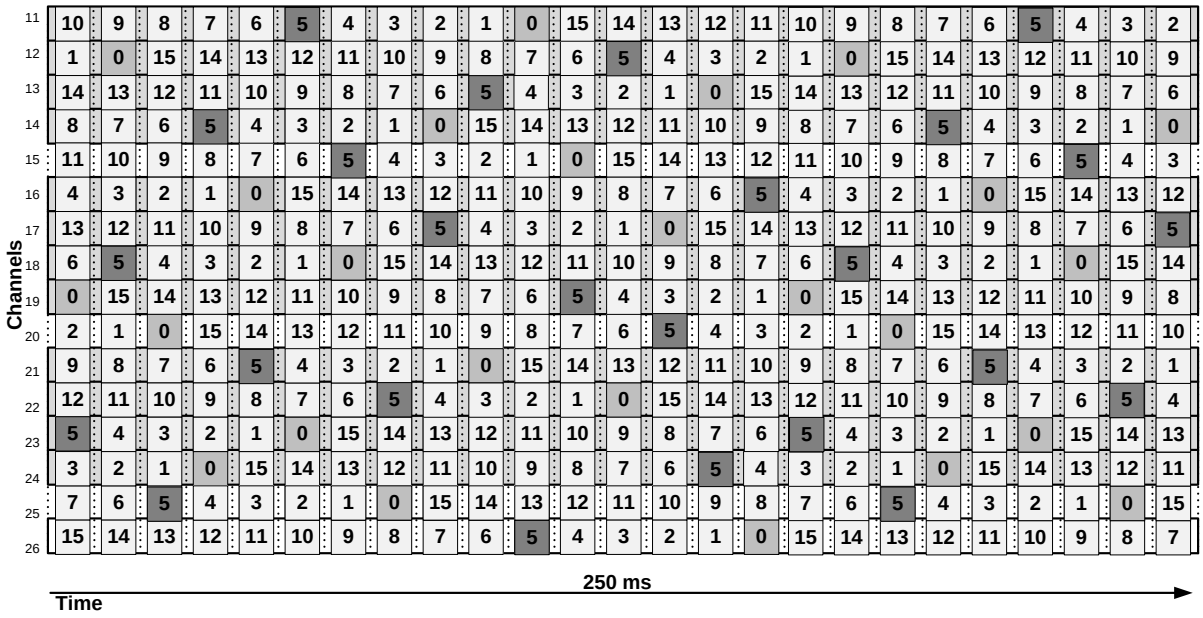
In Figure 61, boxes numbered 0 represent a group of DLEs using predefined channel-hopping-pattern1; its repeating channel-hopping-pattern (using the nomenclature of IEEE 802.15.4) is:

19, 12, 20, 24, 16, 23, 18, 25, 14, 21, 11, 15, 22, 17, 13, 26

Boxes numbered 5 in Figure 61 represent another group of DLEs using channel-hopping-pattern1 with a hopping-pattern offset of 5. A channel-hopping-pattern-offset of 5 has the effect of essentially rotating the channel-hopping sequence to the left by 5, resulting in a repeating channel-hopping sequence (using the nomenclature of IEEE 802.15.4) of:

23, 18, 25, 14, 21, 11, 15, 22, 17, 13, 26, 19, 12, 20, 24, 16

Figure 62 extends this principle, illustrating how different channel-hopping-pattern-offsets may be used for a larger number of DLEs.



NOTE Channel numbers shown are those of IEEE 802.15.4, rather than those of this standard.

Figure 62 – Interleaved channel-hopping-pattern1 with sixteen different channel-hopping-pattern-offsets

In Figure 62, sixteen channels are available. Therefore, up to sixteen DLEs may use channel-hopping-pattern1, each with a different channel-hopping-pattern-offset of 0..15.

As illustrated in Figure 62, a given channel-hopping-pattern may be used concurrently by assigning different channel-hopping-pattern-offsets to various DLEs, superframes (see 9.1.8), or groups of DLEs. As a simple example, each of two DLE clusters may use the same channel-hopping-pattern with different channel-hopping-pattern-offsets, so that the two clusters can share the same radio spectrum without mutual interference. The clusters may be in the same D-subnet or in different D-subnets; as long as they accurately share a consistent timeslot duration and synchronized sense of time, their channel-hopping patterns can be interleaved as shown in Figure 62.

The five predefined channel-hopping-patterns shall exist in every DLE compliant with this standard. This enables routers to advertise concisely the channel-hopping-pattern that is being used and the current channel-hopping-pattern-offset, when it is one of these patterns. This standard also supports customized channel-hopping-patterns in every DLE, in addition to the five predefined patterns, so that the system manager can configure additional channel-hopping-patterns for use by already-joined DLEs.

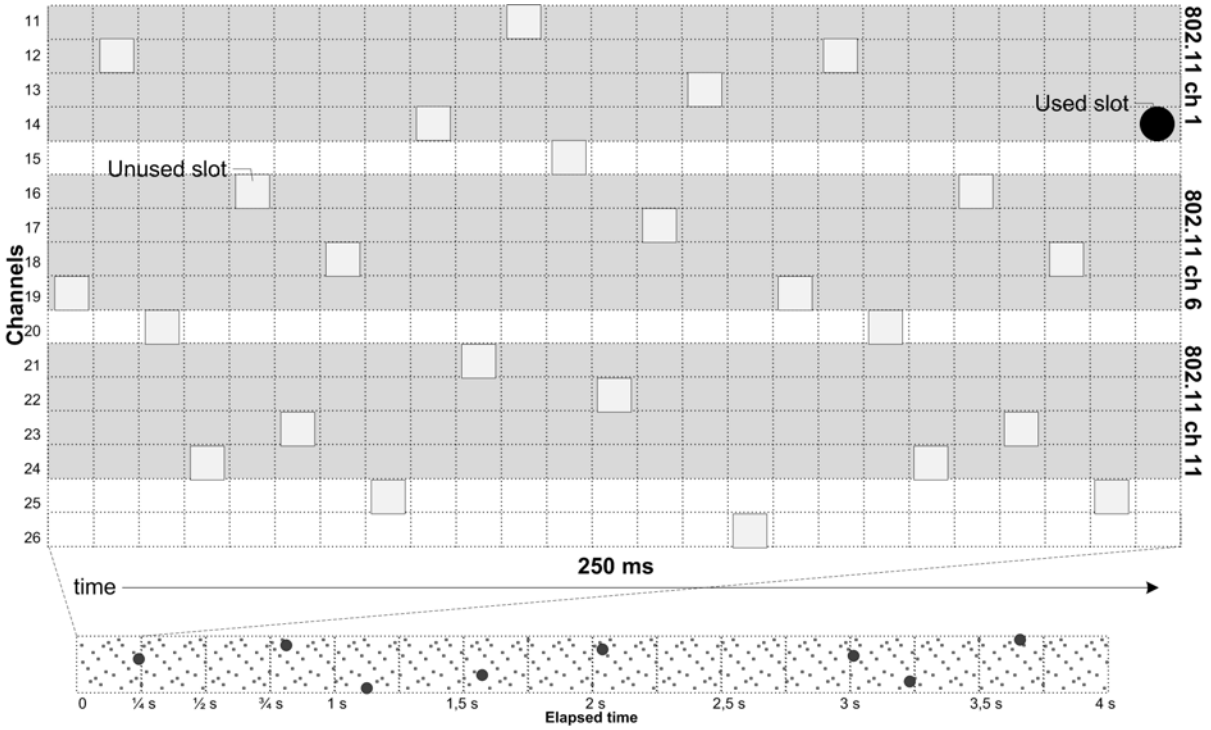
Each predefined channel-hopping-pattern is the same size as the number of channels being used. Thus, for example, channel-hopping-pattern1 uses 16 channels and is 16 hops long. This property allows the channel-hopping-pattern to be interleaved at different offsets as shown in Figure 62.

NOTE 4 Since some DLEs might not support channel 26, which is optional, systems often limit operation to 15 channels (11..25) with essentially the same result.

9.1.7.2.6 Timeslot and channel use

A system shall use slotted-channel-hopping, slow-channel-hopping, or a hybrid combination of the two.

Figure 63 illustrates the use of slotted-channel-hopping. Each timeslot is used with the next successive channel in the channel-hopping pattern.

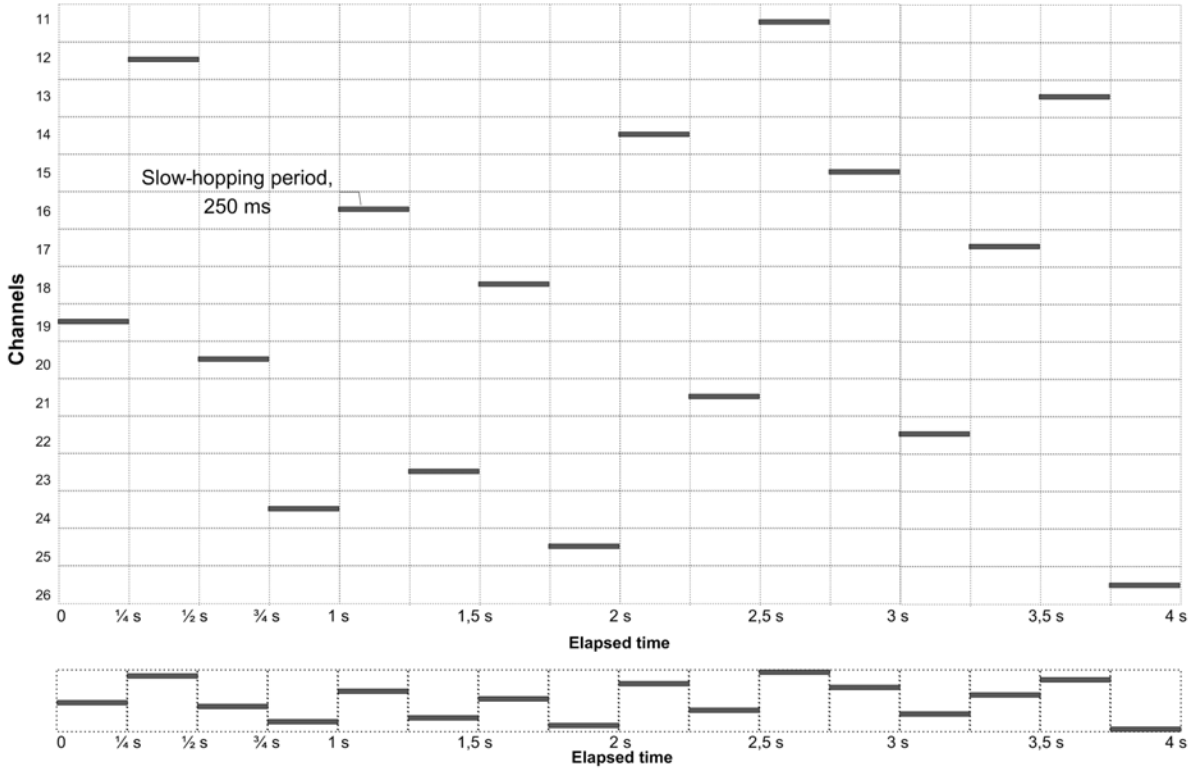


NOTE Channel numbers shown are those of IEEE 802.11 and IEEE 802.15.4, rather than those of this standard.

Figure 63 – Example timeslot allocation for slotted-channel-hopping

The bottom portion of Figure 63 illustrates that the channel-hopping-pattern can be used repeatedly as time progresses. Superframe size and timeslot usage by the DLE is not necessarily tied to lower cyclical DL constructs such as channel-hopping-patterns or 250 ms timeslot alignment intervals. (See 9.1.9.1.3 for a discussion of alignment intervals.)

Figure 64 illustrates the use of slow-channel-hopping. Each channel is used over multiple timeslots.

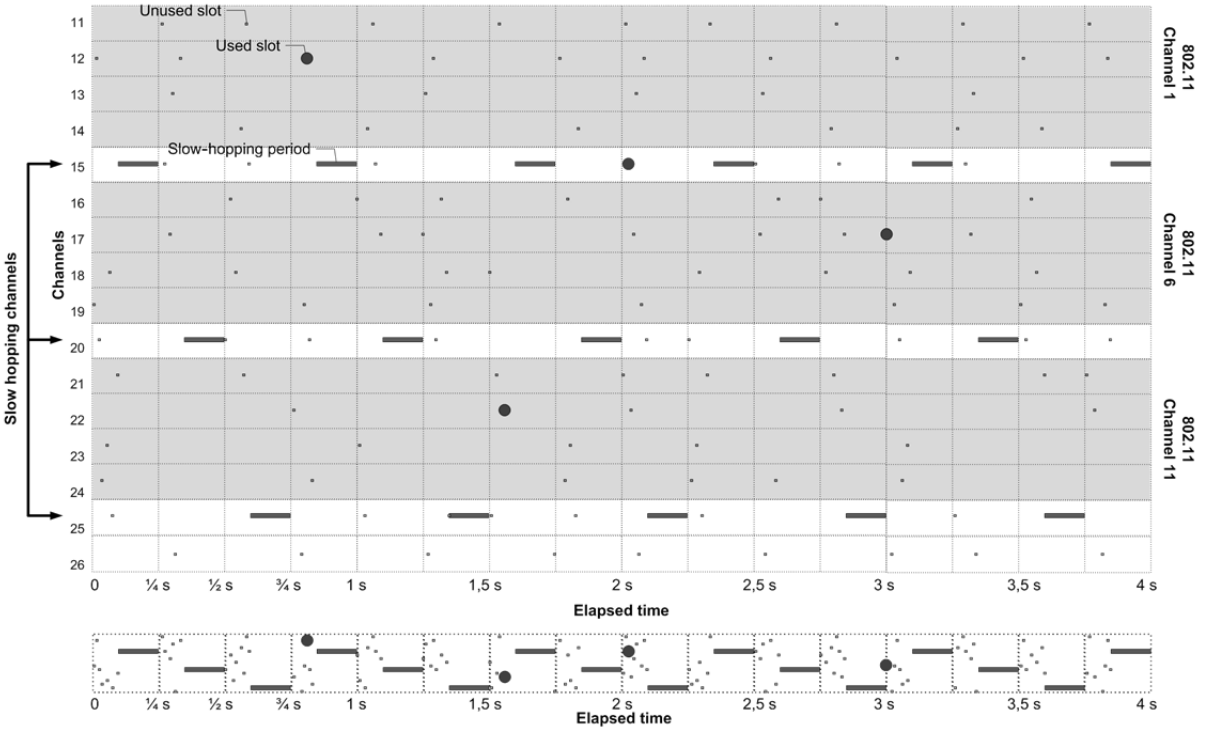


NOTE Channel numbers shown are those of IEEE 802.15.4, rather than those of this standard.

Figure 64 – Example timeslot allocation for slow-channel-hopping

Slow-channel-hopping periods can span a 250 ms timeslot alignment interval. (See 9.1.9.1.3 for a discussion of timeslot alignment intervals.) In such cases, slow-channel-hopping-periods are not interrupted by idle periods, that is, if a slow-channel-hopping-period traverses the edge of a timeslot alignment interval, the radio does not turn off during the otherwise-required idle period.

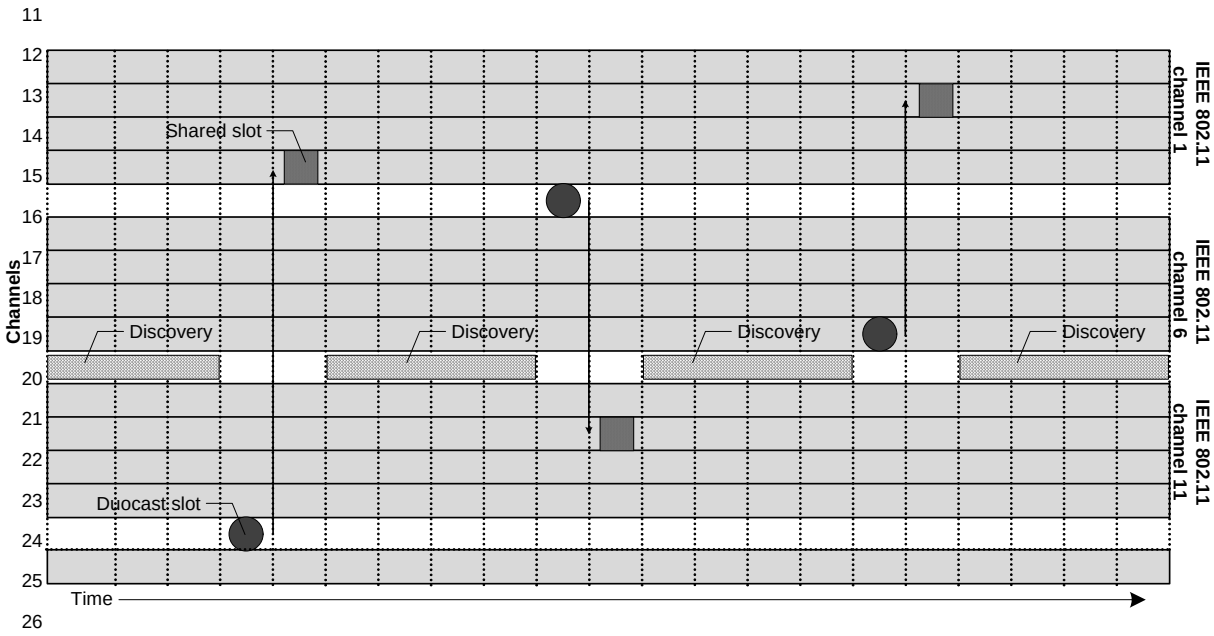
Figure 65 illustrates a hybrid system that combines slotted-channel-hopping and slow-channel-hopping. In this example, within each 250 ms alignment interval, a number of timeslots, each assigned to a different channel, are followed by a slow-channel-hopping-period on a single channel.



NOTE Channel numbers shown are those of IEEE 802.11 and IEEE 802.15.4, rather than those of this standard.

Figure 65 – Hybrid mode with slotted-channel-hopping and slow-channel-hopping

The order in which slotted-channel-hopping and slow-channel-hopping can be combined is flexible; slow-channel-hopping periods need not follow slotted-channel-hopping timeslots. Rather, the two may be used in any sensible combination. For example, Figure 66 shows an example configuration, where a DLE switches between slow-channel-hopping and slotted-channel-hopping.



NOTE Channel numbers shown are those of IEEE 802.11 and IEEE 802.15.4, rather than those of this standard.

Figure 66 – Combining slow-channel-hopping and slotted-channel-hopping

In the example of Figure 66, slotted-channel-hopping is used when broadcast/multicast, duocast/N-cast, or contention-based communication timeslots are allocated explicitly. When a DLE does not have a timeslot allocation, it listens on channel 20, which facilitates neighbor discovery. (See 9.1.9.4.7 for a discussion of duocast/N-cast.)

9.1.8 Superframes

9.1.8.1 General

A superframe is a repeating sequence of timeslots. The number of timeslots in each superframe cycle (its size) and the duration of each of those timeslots determines the period of the superframe cycle. This establishes the structure of the communication schedule for DLEs that use the superframe. For example, a superframe that cycles every 500 ms will allow each DLE that uses a single timeslot within the superframe to communicate every 500 ms.

When a superframe is created, it is given a superframe ID. Figure 67 shows how DLEs may communicate in an example of a three-timeslot superframe.

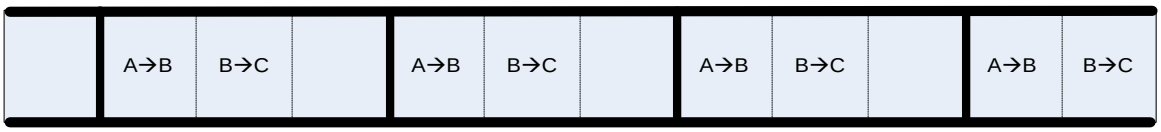


Figure 67 – Example of a three-timeslot superframe and how it repeats

In Figure 67, DLE A may communicate with DLE B during the first timeslot of each superframe cycle. DLE B may communicate with DLE C during the second timeslot of each cycle. The third timeslot of each cycle is unassigned (idle). The cycle repeats every three timeslots.

Figure 67 shows timing cycles and communication links within the same structure, which is a conceptual view. Superframe cycles and communication links are represented as separate but related configurable objects within the DLE. Figure 68 illustrates this data structure, clarifying the distinction between superframes and links, for the same three-timeslot superframe as in Figure 67.

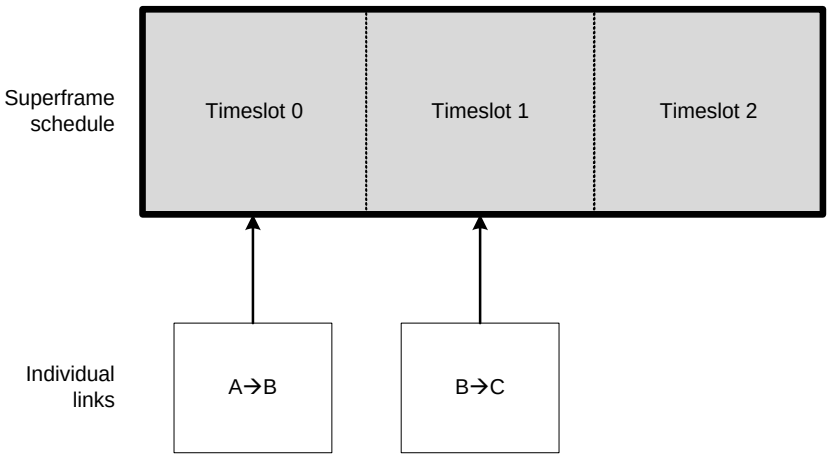


Figure 68 – Superframes and links

As shown in Figure 68, superframes refer to a collection of timeslots. Links refer to the use of superframe timeslots for communication between a specific pair of DLEs. A timeslot is a period of time. A superframe is a cyclic schedule of timeslots and associated channels. A link describes a specific activity that repeats within a superframe cyclic schedule.

The system manager configures matching sets of links among a collection of DLEs that communicate with each other. For example, a link on DLE A may be configured to transmit Data DPDUs to DLE B on a particular superframe cycle. On the same cycle, DLE B should have a link that is configured to listen for an incoming Data DPDU. These two links are matched in the sense that the system manager has configured these two related operations to occur concurrently on the same channel.

Several performance parameters are determined by superframe period and how links are assigned to superframes. In general, shorter-period superframes result in lower Data DPDU latency and increased digital bandwidth, at the expense of increased energy consumption and more concentrated allocation of digital bandwidth. Longer-period superframes generally result in higher latency and lower digital bandwidth, but with reduced energy consumption and less concentrated allocation of digital bandwidth. These tradeoffs should be carefully considered when determining superframe period and link density within a superframe.

A given DLE may be configured to use concurrently several superframes of different sizes. A link with one timeslot within a superframe of length L slots repeats twice as frequently as a similar one-timeslot link within a superframe of length $2 \times L$ slots, thus allowing for twice the throughput per second.

A DLE may use more than one superframe simultaneously. Also, not all DLEs in a D-subnet need to participate in each superframe. By configuring a DLE to participate in multiple concurrent superframes of different lengths, it is possible to establish multiple communication schedules with incommensurate periods that all operate simultaneously.

Superframes are numbered for identification, but these superframe numbers are limited in scope to the DLE where the superframe is used. Since the scope of a superframe number is a single DLE, a neighboring DLE can use the same superframe number for a completely different purpose. Superframes can be added, removed, activated, and deactivated while the D-subnet is running.

Figure 69 shows how timeslots in different superframes are aligned, even though the superframes may cycle at independent rates.

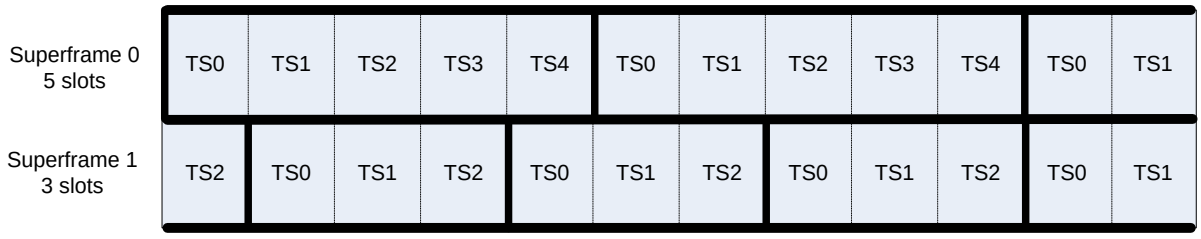


Figure 69 – Multiple superframes with aligned timeslots

NOTE Timeslot alignment is a result of defining all slots to have identical durations and of realigning timeslots to TAI time every 250 ms (see 9.1.9.1.3). Superframes with timeslots of different durations are usable simultaneously within a D-subnet. However, the designers of this standard considered only configurations wherein a single timeslot duration is used during operation of a given D-subnet (changeable at D-subnet reinitialization).

A DLE with multiple links in a timeslot may encounter link collisions when two or more links coincide. To address such situations, each link is assigned a priority. A higher-numbered link priority means the link takes precedence over a link with a lower-numbered priority. In the event of a link collision, the link with the higher-numbered priority is used. If two links have the same priority, a superframe priority is used. See 9.1.8.5.

In addition to link priority, each Data DPDU is assigned a priority by its originating DLE. Once a link is selected based on link priority, the priority of the Data DPDU is used to weight the relative importance of all queued Data DPDUs that can use the link.

9.1.8.2 Exponential backoff

Links may be shared or dedicated. On the receiver side, there is no structural difference between shared versus dedicated links. On the transaction initiator side, an exponential-backoff bit in the link configuration specification indicates whether the link is shared or dedicated. If a link is shared, as indicated by the link's exponential-backoff bit being set (to 1), the transaction initiator shall use exponential backoff for retries using that link. If a link is dedicated, as indicated by the link's exponential-backoff bit being reset (to 0), the transaction initiator shall not use exponential backoff in that link.

It is possible, and sometimes reasonable, for a system manager to configure multiple DLEs to transmit at the same time and on the same radio channel, without setting an exponential-backoff bit in the applicable links. The term "dedicated link" is used merely to indicate that exponential backoff is not applied to such links.

Exponential backoff shall be applied when, and only when, a DLE transmits a unicast Data DPDU on a shared link and does not receive an error-free ACK/NAK DPDU, which implies a possible collision. A unicast transmission that is aborted due to CCA sensing shall be treated as equivalent to an unsuccessful transmission in the context of exponential backoff. Exponential backoff is intended to resolve such collisions. Exponential backoff shall operate on a per-neighbor basis, and apply to all Data DPDUs in the message queue addressed to that neighbor, regardless of Data DPDU priority.

For each neighbor, the DLE maintains a backoff exponent and a backoff counter, called BackoffExponent[Neighbor] and BackoffCounter[Neighbor] herein. BackoffExponent[ ] and BackoffCounter[ ] are inaccessible implementation internals, and therefore are not included in the DL object model.

BackoffExponent[Neighbor] and BackoffCounter[Neighbor] are set to zero every time an ACK/NAK DPDU is received for a unicast Data DPDU that was sent to a particular neighbor in a shared link.

A BackoffCounter[Neighbor] value of zero allows the DLE to send a Data DPDU at the next shared-link transmission opportunity. Following an unsuccessful transmission to the neighbor in a shared link, if the current value of BackoffExponent[Neighbor] is less than dlmo.MaxBackoffExp, the DLE increments BackoffExponent[Neighbor] and then sets BackoffCounter[Neighbor] by selecting a value uniformly from the interval $0..2^{(\text{BackoffExponent[Neighbor]})}-1$. For each transmit opportunity in a shared link, BackoffCounter[Neighbor] is decremented until it reaches zero. If a transmit opportunity is in a dedicated link (no exponential backoff indicator), the DLE may use the link regardless of the value of BackoffCounter[Neighbor]. The attribute dlmo.MaxBackoffExp limits the maximum value of BackoffExponent[Neighbor].

Retry behavior can be configured by the system manager. DLMO attributes that relate to retries include:

- dlmo.MaxBackoffExp. The maximum value for BackoffExponent[Neighbor].
- dlmo.MaxLifetime and dlmo.Graph.MaxLifetime: maximum lifetime of a Data DPDU. A Data DPDU that is being forwarded shall be deleted if held in a DLE's message queue for longer than MaxLifetime. dlmo.MaxLifetime provides a default value for the DLE. A non-null value for dlmo.Graph[ ].MaxLifetime indicates that dlmo.MaxLifetime shall be overridden and set to the specified value for Data DPDUs following that particular graph.
- Operation of exponential backoff that is illustrated in the following pseudocode:

```
// For each neighbor, independently
BExp[Nei] = 0; // BackoffExponent[Neighbor] in text
BCnt[Nei] = 0; // BackoffCounter[Neighbor] in text
For each timeslot (
If (transmit link and Data DPDU match)( // See 9.1.8.5
   If (not exponential backoff link) (
      // Dedicated link
      Attempt to transmit Data DPDU using link;
      If (transmit was successful) remove Data DPDU from queue;
      )
   Else (
      // Shared link
      If (BCnt[Nei] > 0) BCnt[Nei]--
      Else (
         Attempt to transmit Data DPDU in link;
         If (transmit was successful) (
            Remove Data DPDU from message queue;
            BExp[Nei]=0;
            BCnt[Nei]=0;
            )
         Else (
            // Transmit failed; exponential backoff
            If (BExp[Nei] < MaxBackoffExp) BExp[Nei]++;
               BCnt[Nei] = Random (0, 2^(BExp[Nei]-1));
            )
         )
      )
   )
Delete all messages beyond MaxLifetime;
If (no queued Data DPDU for neighbor) (BCnt[Nei]=0; BExp[Nei]=0;)
```

NOTE As described in 9.1.8.5, it is possible for a link to be configured as a transmit/receive (T/R) link, which is a compressed representation of a paired transmit link and receive link. Logically, T/R links are processed as two independent links.

9.1.8.3 Superframe channel use

Timeslots within a superframe are associated with a slow or slotted-channel-hopping pattern, as well as an offset into that pattern.

From the perspective of each DLE using a superframe, there is a baseline channel-hopping pattern offset, which may vary from DLE to DLE and which may be overridden with an alternative offset applied to a link or collection of links within a superframe.

A given unicast D-transaction occurs on a single channel, with the Data DPDU and ACK/NAK DPDU(s) all transmitted on the same channel.

A superframe is not limited to one channel at a time; rather, a superframe is a two-dimensional structure indicating time and channel, as was previously illustrated in Figure 62 (see 9.1.7.2.5).

Figure 62 shows a superframe, with time on the horizontal axis and channel on the vertical axis. The superframe spans all of the channels over the length (duration) of that superframe. As shown in Figure 62, sixteen DLEs may use sixteen different offsets from channel-hopping pattern A; the superframe encompasses all of the channel assignments for all of the superframe timeslots.

The default channel offset may be different for transmitting versus receiving and may vary by link.

The period of the channel-hopping pattern is not necessarily related to the length of the superframe. Referring to Figure 62, a superframe might be configured as 25 timeslots long, even though channel-hopping pattern A is only 16 hops long.

For frequency diversity, superframe length and channel-hopping pattern size may be configured to be relatively prime, that is, with no common factors. As a counter-example, consider a configuration wherein superframe length is 25 timeslots, with a channel-hopping

pattern repeating on a 15-channel cycle, resulting in a superframe schedule where only 3 of the 15 available channels are ever used. Such an arrangement can cause regulatory issues in situations where use of all channels is required by each device.

9.1.8.4 Organizing superframes

9.1.8.4.1 General

Two general superframe types are supported:

- Slotted-channel-hopping, which makes optimal use of available digital bandwidth and supports battery-powered routers.
- Slow-channel-hopping, intended for routers with available energy to run their receivers continuously during a given period. Slow-channel-hopping allows neighboring DLEs to operate with less exacting time synchronization requirements, particularly during the neighbor discovery process.

Hybrid configurations may be arranged by combining superframes, for example, one slotted and one slow. Slotted- and slow-channel-hopping will be discussed separately, followed by some examples of hybrid configurations.

NOTE In the marketplace, slow-channel-hopping is sometimes referred to as CSMA, and slotted-channel-hopping as TDMA. These terms are not used in this standard, except to the extent that CSMA/CA is supported by the standard in a literal sense. (See 9.1.9.4.8.) Slow-channel-hopping is built on a TDMA base, slotted-channel-hopping includes CSMA aspects. The solution designer is free to mix the approaches.

9.1.8.4.2 Superframe scope

Superframes are commonly discussed as abstractions that span several DLEs. Nonetheless, while the superframe may be conceptualized at the D-subnet level, the scope of the superframe data structure is limited to each DLE. A superframe is instantiated as a data structure on a single DLE that independently drives its DLE state machine. A DLE's superframe definitions need to relate to those of its neighbors so that DLEs communicate at the same time. Superframe definitions within each DLE are numbered, but that numbering is only needed by the DMAP for table read/write operations and by other objects and attributes within the DLE that refer to the superframe.

A superframe may be contrasted with a routing graph's scope. A graph ID, unlike a superframe ID, is carried in a Data DPDU's DROUT header, and a graph ID shall be consistent and unique in all DLEs that use the graph. No such constraints apply to a superframe.

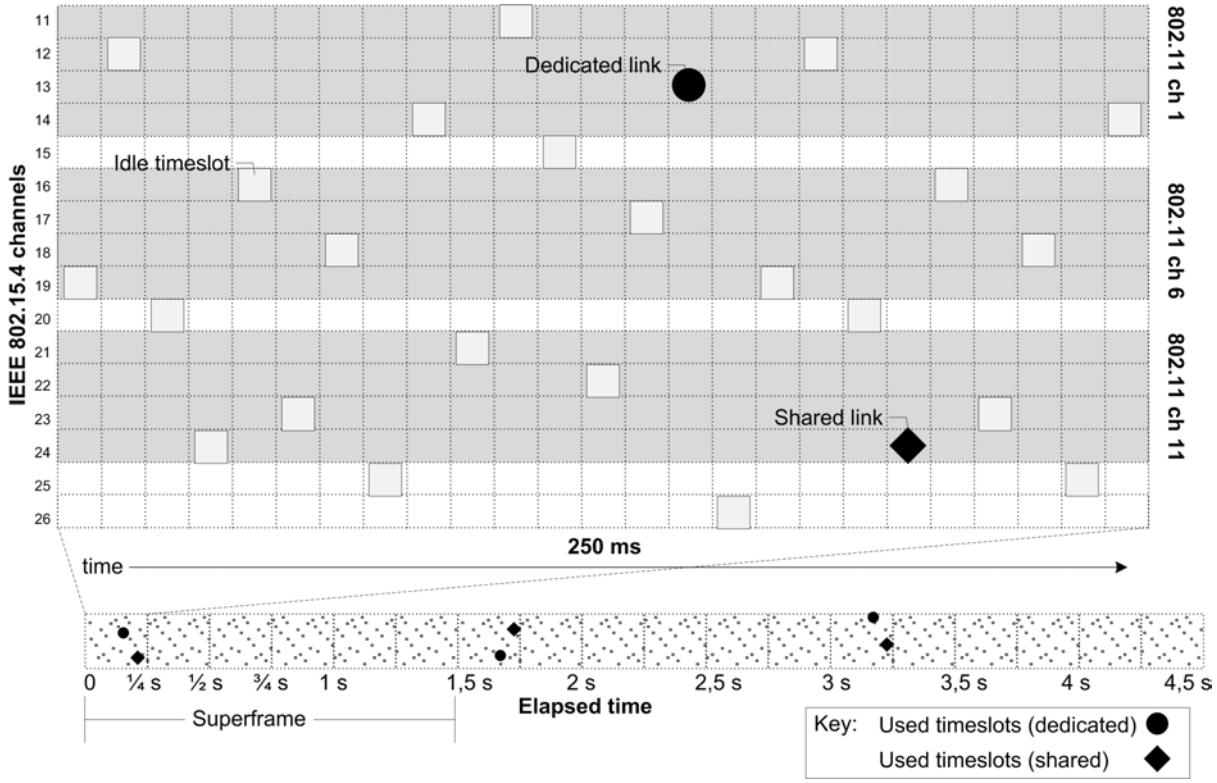
9.1.8.4.3 Blocks of contention-based capacity

Mains powered routers may sensibly operate their receivers continuously. As its lowest priority superframe, such a router may support a superframe comprised partially or entirely of receive links. The router's neighbors may maintain corresponding shared transmit links to the router. Such a configuration results in blocks of contention-based digital bandwidth available to the routers' neighbors. Slow-channel-hopping or slotted-channel-hopping may be used in a superframe of that type. Channel-hopping-offset may be selected to avoid collisions between dedicated links versus a general inventory of shared links.

9.1.8.4.4 Slotted-channel-hopping

Slotted-channel-hopping uses channel-hopping superframe timeslots of equal duration. Each superframe timeslot uses a different radio channel in a hopping pattern. In slotted-channel-hopping, each superframe timeslot is intended to accommodate one D-transaction, including a Data DPDU and its ACK/NAK DPDU(s).

Figure 70 illustrates a slotted-channel-hopping superframe from the perspective of one DLE, which may be a router.



NOTE Channel numbers shown are those of IEEE 802.11 and IEEE 802.15.4, rather than those of this standard.

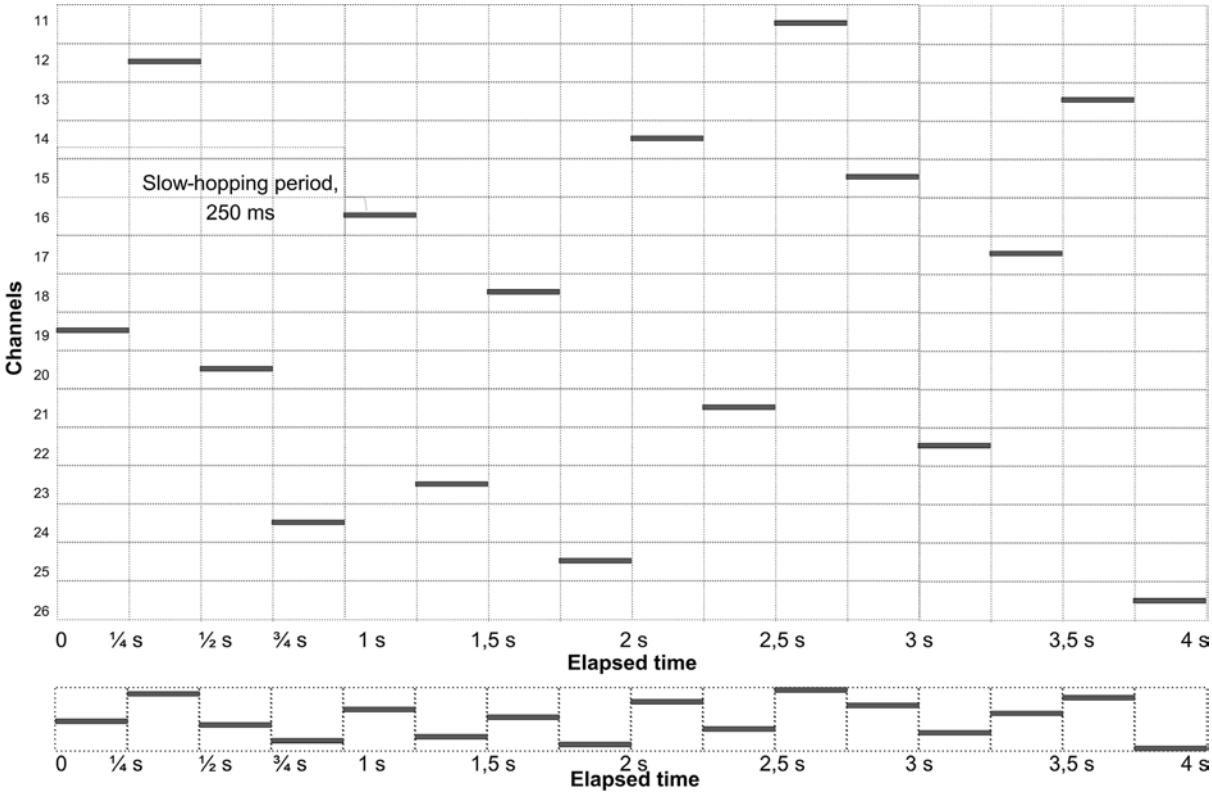
Figure 70 – Example superframe for slotted-channel-hopping

In this example, the superframe is 1,5 s long. Timeslots with link assignments are depicted with circles (dedicated links) and diamonds (shared links). As Figure 70 shows, from the perspective of a single DLE, many superframe timeslots might be left idle. Timeslots with link assignments repeat at a fixed interval defined by the superframe length.

9.1.8.4.5 Slow-channel-hopping

In slow-channel-hopping, a collection of contiguous superframe timeslots is grouped on a single radio channel. Each such collection of superframe timeslots is treated as a single slow-channel-hopping period.

Figure 71 illustrates slow-channel-hopping.



NOTE Channel numbers shown are those of IEEE 802.15.4, rather than those of this standard.

Figure 71 – Example superframe for slow-channel-hopping

Timeslots in a slow-channel-hopping superframe are generally shared, providing immediate, contention-based channel bandwidth on demand to a router's immediate neighbors.

Figure 72 shows the main components of a slow-channel-hopping superframe.

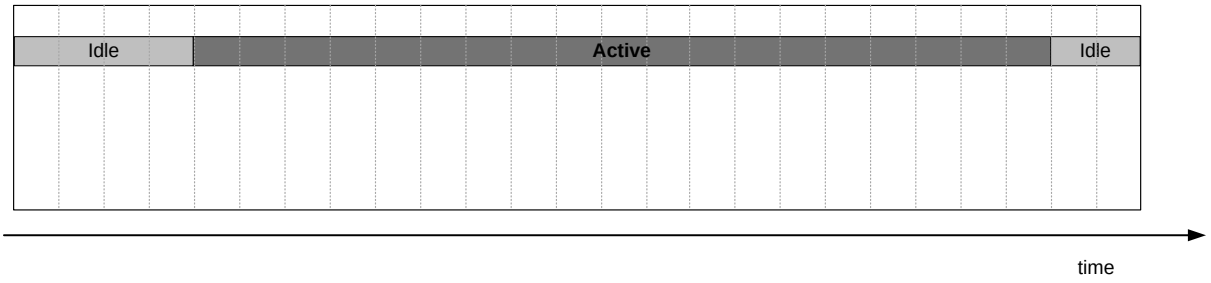
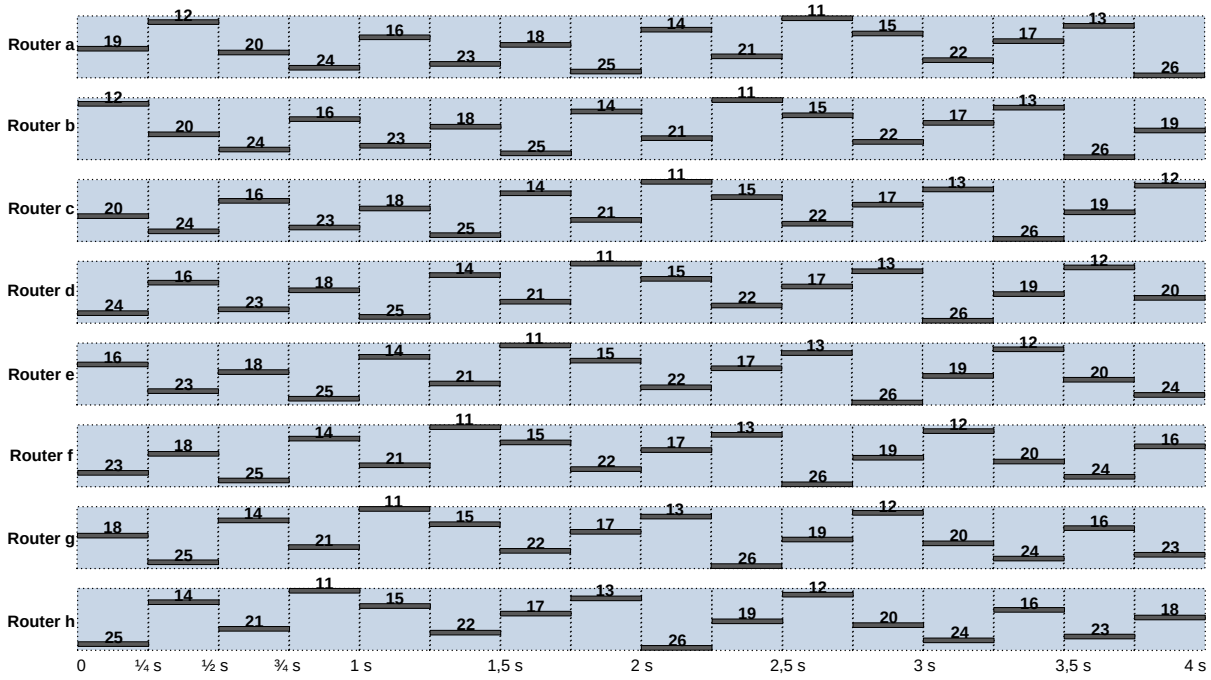


Figure 72 – Components of a slow-channel-hopping superframe

A baseline slow-channel-hopping period has a fixed duration, with a fixed number of idle timeslots (which may be zero) at the beginning and end of the hop. The example of a slow-channel-hopping period shown in Figure 72 is comprised of 25 timeslots, including four idle timeslots at the beginning, nineteen active timeslots in the middle, and two idle timeslots at the end. The idle timeslots are intended to support hybrid configurations where slow-channel-hopping superframes are paired with slotted-channel-hopping superframes, with the slotted-channel-hopping timeslots scheduled for use during the idle periods of the slow-channel-hopping superframe.

NOTE 1 Idle periods as described here are configurable by matching superframe and channel-hopping phases, and defining links that match the desired active range.

It is not necessary for all routers in a common area to hop together. Figure 73 shows how many routers can be assigned slow-hopping patterns that are disjoint from each other, thus avoiding collisions. It does not imply that all routers in a system use disjoint communication channels.



NOTE Channel numbers shown are those of IEEE 802.15.4, rather than those of this standard.

Figure 73 – Example configuration for avoiding collisions among routers

Each router may use a different offset into the channel-hopping pattern. With a 16-channel hopping pattern, each of up to 16 routers may be configured with a different offset into the pattern, so that no two routers use the same channel at the same time.

NOTE 2 Although Figure 73 purports to show how collisions can be avoided through disjoint assignments of channel-hopping patterns, a realistic system would require at least one shared channel-hopping pattern via which the routers could communicate with each other.

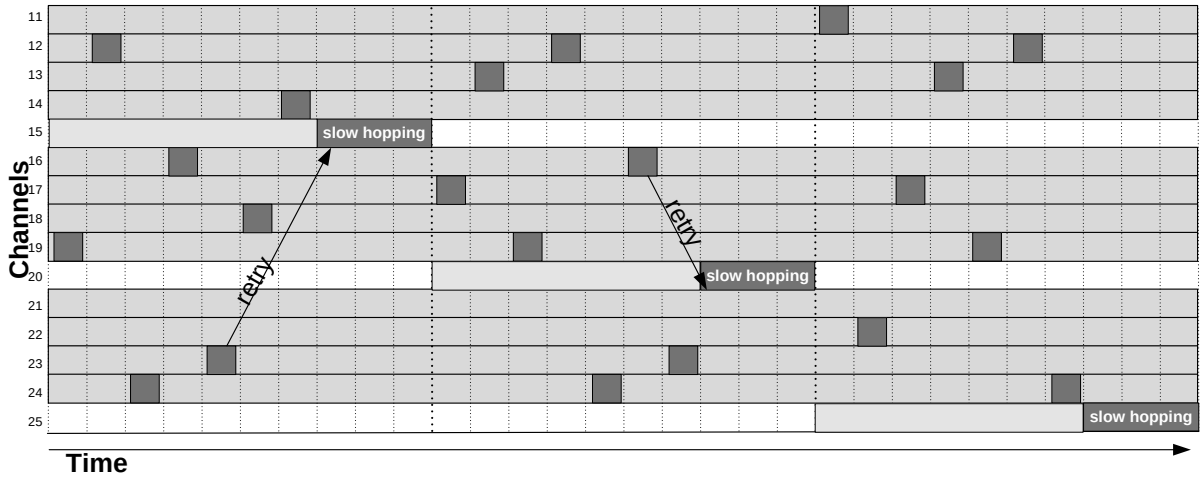
See 9.4.3.5.5 for more detail on slow-channel-hopping.

9.1.8.4.6 Hybrid channel-hopping configurations

Hybrid configurations may use combinations of the slotted-channel-hopping and slow-channel-hopping superframes.

Hybrid configurations are usually arranged so that slotted-channel-hopping links are allocated for scheduled, periodic messaging. This may leave blocks of lightly-used slow-channel-hopping capacity, available on a contention basis, for less predictable uses such as alarms and retries.

The example in Figure 74 illustrates how slotted-channel-hopping and slow-channel-hopping superframes may be combined.



NOTE Channel numbers shown are those of IEEE 802.15.4, rather than those of this standard.

Figure 74 – Hybrid configuration

In Figure 74, slotted-channel-hopping has been overlaid on a slow-channel-hopping background. The slow-hopping periods fill in the time between periodic collections of dedicated superframe timeslots.

In this configuration, if an attempted transmission in a dedicated timeslot fails, the succeeding slow-channel-hopping period can be used to retry the transmission. In Figure 74, two of the superframe timeslots are shown with retries on different channels during the subsequent slow-channel-hopping period.

9.1.8.4.7 Superframes and spectrum management

The term spectrum management refers to the ability of the system manager to configure a D-subnet to block unwanted channels from operation in a D-subnet.

Each superframe includes a channel map, called Superframe[ ].ChMap, that is a bit mask of channels that shall be included and excluded in the hop sequence that is referenced by the superframe. Excluded channels have the effect of shortening the hop sequence.

For example, a superframe channel map that includes channels 11..25 and excludes channel 26 has the effect of shortening the hop sequence by removing channel 26 from the hop sequence. More generally, the system manager may eliminate any collection of channels from the hop sequences that are referenced by superframes in a D-subnet, with the result of removing those channels from operation.

There is also a channel map in the DeviceCapability attribute that is reported to the system manager when the DLE joins the D-subnet. The DeviceCapability channel map does not shorten any channel-hopping sequence used by the DLE, but rather is a signal to the system manager that, for regulatory reasons, any link using one of the excluded channels will be treated as idle.

The system manager may also block use of certain channels through the attribute dlmo.IdleChannels. Unlike the channel map in the superframes, dlmo.IdleChannels does not cause hop sequences to be shortened; rather, it causes links on designated channels to be treated as idle. dlmo.IdleChannels is intended to provide a quick way for the system manager to disable certain channels in a way that does not require D-subnet-wide coordination of revised hop sequences.

Channel-specific diagnostics, as described in 9.4.2.27, provide the system manager with information to support spectrum management.

9.1.8.5 DLE message queue operation

DL routers compliant with this standard shall support a DLE message queue. This message queue has some attributes that can be configured by the system manager, affecting how the queue operates.

The standard does not generally specify internal DLE mechanisms. However, to a limited degree, the DLE message queue is specified by the standard. The system manager can configure the DLE message queue to achieve a particular quality of service objectives, and a limited model of message queue behavior is implicit in the configuration alternatives provided to the system manager.

When a DLE receives a unicast Data DPDU from its neighbor, it first assesses whether the DSDU should be passed to the NL or forwarded to another DLE through the DL, as described in 9.3.3.6.

If the Data DPDU needs to be forwarded, the DLE then evaluates whether the Data DPDU should be accepted or NAKed, in part based on the available capacity of the DLE message queue. Data DPDUs shall be NAKed if the DLE message queue has run out of capacity for Data DPDUs of that type.

The DL reports its forwarding queue capacity to the system manager when the DLE joins the D-subnet, through the attribute dlmo.DeviceCapability, field QueueCapacity. This externally reported queue capacity does not include portions of the queue that are reserved for the DLE's internal use. For example, the DLE message queue in a particular DLE might report that it has a queue capacity for five Data DPDUs. The system manager is then able to configure those five positions in the queue. This nominal capacity refers only to a portion of the message queue that is exclusively used to route Data DPDUs through the DLE. In practice, the actual DLE has additional message queue capacity space that it does not report to the system manager, because the DLE also needs to handle Data DPDUs on its own behalf. Unreported message buffer capacity, for the DLE's own use, is considered an internal DLE matter and is not allocated by the system manager. The system manager assumes that the DLE has sufficient message queue capacity for its own use when contracts are granted.

Consider the example of a field router with a reported DLE message buffer capacity of eight Data DPDUs. The actual buffer capacity may be twelve Data DPDUs, in which case the difference between reported and actual capacity (four Data DPDUs) is for the DLE's own use. In this example, the system manager might reasonably configure the DLE's nominal buffer capacity as follows:

- No more than three of the eight buffers will be used to forward Data DPDUs with priority ≤ 2.
- No more than five of the eight buffers may be used to forward Data DPDUs with priority ≤ 5.

See 9.4.2.26 for further discussion of queue buffer capacity and priority levels.

In addition, for a finer grained degree of control, the system manager can designate a certain number of buffers exclusively to forward Data DPDUs that are being routed along a specific graph, as described in 9.4.3.7.

Figure 75 provides an overview of how links interact with an implicit DLE message queue within a DLE.

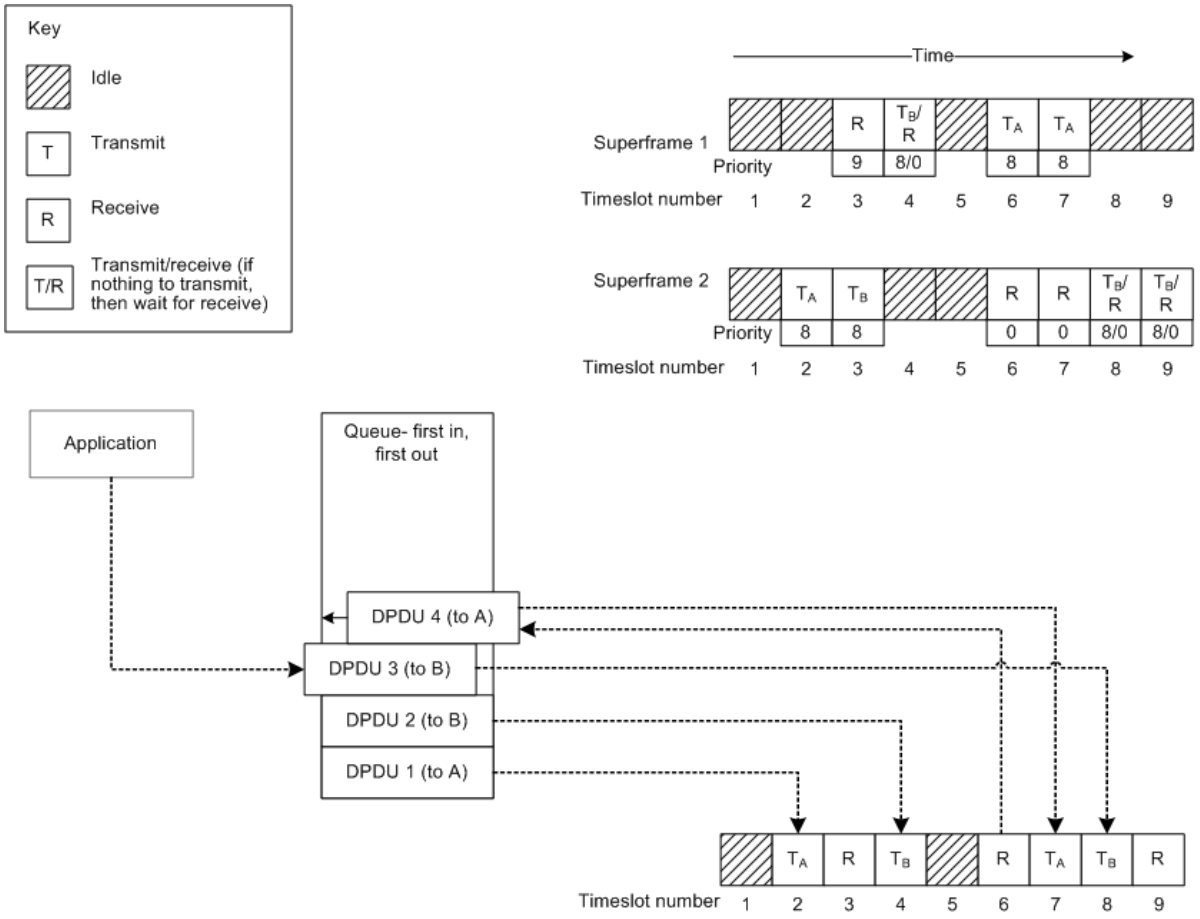


Figure 75 – Timeslot allocation and message queue

As shown in Figure 75, Data DPDUs are held in a message queue until transmit links become available. Data DPDUs are placed in the queue in the order they are received. Generally, retrieval of Data DPDUs from the queue is first in, first out (FIFO).

NOTE This simplified example shows only a single destination address for each Data DPDU. In practice, each Data DPDU on the message queue is actually a candidate for links to multiple neighbors.

Consider the example of Data DPDU 3. Data DPDU 3 originates in an application and enters the DL through the DDSAP (Figure 53). When the Data DPDU is processed by the DLE, it is placed in the DLE's message queue. Based on the Data DPDU's ultimate destination, the DLE determines that the Data DPDU needs a link to DLE B for its next hop. Data DPDUs 1 and 2 are already on the queue, so Data DPDU 3 is queued behind them. When a link to DLE B becomes available, Data DPDU 2 will be sent before Data DPDU 3.

Each Data DPDU on the queue is assigned a priority by the originating DLE. This simplified tutorial assumes that all Data DPDUs have equal priority. In actual practice, link priority takes precedence over Data DPDU priority, in the sense that Data DPDUs are not considered for transmission until after a transmit link is selected. Once a transmit link has been selected, the Data DPDU on the queue is selected based first on priority, and then on a FIFO basis if multiple candidate Data DPDUs have the same priority.

Data DPDU 4 shows an example where a Data DPDU is received in one timeslot and is available to be forwarded in the next timeslot. In general, a Data DPDU received in one timeslot (timeslot N) shall be available on the queue for forwarding in the next timeslot (timeslot N + 1). An exception is allowed when default timeslot template 3 is used (see Figure 157). In that case, the D-transaction might be incomplete at the start of the next timeslot. A Data DPDU received starting in one timeslot (timeslot N), using default timeslot template 3,

shall be available on the queue for forwarding in the timeslot following the next timeslot (timeslot N + 2).

In Figure 75, two superframes are shown, and each superframe is associated with a series of links. For example, timeslot 3 in superframe 1 is associated with a receive link (R), while timeslot 3 in superframe 2 is associated with a transmit link to DLE B (T_B). Idle slots do not have defined links.

Each link has a priority. The attributes dlmo.LinkPriorityXmit and dlmo.LinkPriorityRcv provide default link priorities, which may be overridden for any particular link. The example in Figure 75 mostly shows the default priorities of 0 for receive links and 8 for transmit links.

In most cases the system manager should assign lower priority to receive links (R) than to transmit links (T), thus giving precedence to servicing outgoing Data DPDUs on its queue. However, this is not a strict requirement. For example, if a latency-critical incoming flow is scheduled for a particular timeslot, the system manager may configure receive links with higher priority in that case. As an illustration in Figure 75, the third link in superframe 1 is assigned a priority of 9, giving this particular receive link a higher priority than a transmit link at the same time.

Transmit/receive (T/R) timeslots use a compressed format to combine a transmit link and a receive link. Logically, a T/R link is two links, with the receive part of the link having a priority of dlmo.LinkPriorityRcv which defaults to zero. For example, if a timeslot has a high priority T_A/R link and lower priority T_B link, the T_B link has higher priority than the R part of the T_A/R link. Baseline operation is that a Data DPDU queued for transmission and a link are matched at the start of a timeslot, and the timeslot is assigned to a D-transaction that will be run to completion according to the link configuration. In the event that a D-transaction is aborted due to CCA detection of competing channel activity, an optimized implementation may complete the timeslot using a receive link that is valid for the same time interval.

Once a queued Data DPDU has been transmitted and acknowledged, the D-transaction is deemed successful and the Data DPDU is deleted from the queue. The following example assumes that all transactions are successful.

The box at the bottom right of Figure 75 illustrates how links are used in the following example.

- Timeslot 1: The first link in both superframes 1 and 2 are idle; therefore, the first timeslot is idle.
- Timeslot 2: The second link in superframe 1 is idle, but there is a transmit link for DLE A (T_A) in superframe 2. There is also a Data DPDU to DLE A in the queue; therefore, the second timeslot is assigned for transmission to DLE A, and Data DPDU 1 is sent.
- Timeslot 3: Superframe 1 has a receive link, and superframe 2 has a transmit link. The link in superframe 1 takes precedence due to its priority, so timeslot 3 is used to listen for incoming Data DPDUs. (As described above, receive links are usually assigned lower priority than transmit links. This example illustrates how a system manager can give priority to a receive link, for example, to service an incoming flow.)
- Timeslot 4: Superframe 1 has a T_B/R link, indicating that it should transmit if there is a Data DPDU for DLE B in the queue. Data DPDU 2 is sent.
- Timeslot 5: Both superframes are idle in timeslot 5, so the timeslot is idle.
- Timeslot 6: Superframe 1 designates a transmission link to DLE A, but there is no longer a Data DPDU for DLE A in the queue. Since there is nothing useful for the transmit link to do in this slot, the link in superframe 2 is used. The DLE receives an inbound Data DPDU, determines that its next hop is to DLE A, and places the Data DPDU on the queue.

- Timeslot 7: Now that there is a Data DPDU for DLE A on the queue, the transmit link in superframe 1 gets priority and Data DPDU 4 is sent. Note that Data DPDU 3 was skipped over because no T_B link has become available yet.
- Timeslot 8: Now that a T_B link is available, Data DPDU 3 is sent.
- Timeslot 9: The T_B/R link results in a receive slot, because there is no Data DPDU to DLE B on the queue.

This example was simplified in some essential respects.

– As noted above, Data DPDU priorities were assumed to be equal. In practice, if two Data DPDUs on a message queue both match a given link, the Data DPDU with the higher priority is transmitted. The FIFO queue position is relevant only for Data DPDUs of equal priority.

– If a unicast Data DPDU does not receive an acknowledgment, it stays on the queue and the DL retry strategy is applied. See 9.1.8.2.

– A link may be configured for use by a particular graph. In that case, Data DPDUs with that graph ID shall be granted prioritized or exclusive access to the link. See 9.4.3.7.

– A DLE may be designed to skip links on radio channels with a history of subpar connectivity. A DLE may also skip links in order to retry on an alternative link. See 9.1.7.2.4.

Operation of queue processing is illustrated in the following pseudocode:

```
For each timeslot (
   Order Data DPDUs on queue by priority;
   // FIFO within priority

   Order links by priority;
   // Treat T/R link as two links, with receive side
   // of link assigned link priority dlmo.LinkPriorityRcv;
   // Within link priority, order by superframe priority,
   // then superframe number (highest first);

   For each link, in priority order
      If it is a receive link (
         Use the receive link;
         Done with timeslot;
         )
      Else // it is a transmit link
         For each Data DPDU, in priority order
            If link matches Data DPDU (
                  Use the link;
                  Done with timeslot;
                  )
)
```

9.1.9 DL time keeping

9.1.9.1 Timing

9.1.9.1.1 General

The DL propagates and uses international atomic time (TAI) for its internal operation, and also provides TAI time as a service (through the DMAP) to wireless devices compliant with this standard.

DLEs within a D-subnet may be configured to track a shared sense of time to within a few milliseconds of each other, to support sequence of event reporting and other application-layer coordinated operations within the scope of a D-subnet. Synchronized time among immediate neighbors is also essential for operation of the wireless protocol.

Slotted-channel-hopping requires tight time synchronization among immediate neighbors. However, the internal clock of a non-routing DLE that has been disconnected from D-subnet operation for more than a few minutes may have drifted by tens or hundreds of milliseconds

or more in relation to the overall D-subnet clock. Slow or hybrid channel-hopping configurations support the continued operation of such DLEs.

9.1.9.1.2 International atomic time

In this standard, time is based on international atomic time (TAI) as the time reference. See 5.6.

9.1.9.1.3 Alignment intervals

In this standard, one-second TAI increments are divided into 250 ms (1/4 s) alignment intervals, wherein the 250 ms (2^{-2} s) cycles shall align with nominal TAI s as shown in Figure 76.

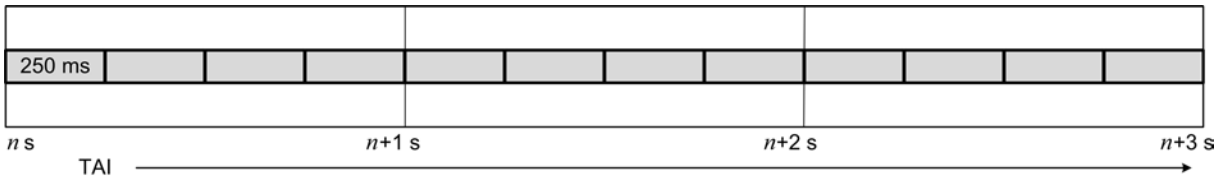


Figure 76 – 250 ms alignment intervals

Continuous control loops in the process industries, which have been a traditional focus of IEC TC 65, are frequently based on fixed-period computation of control outputs. These process loops usually repeat at rates of 4 Hz (250 ms), 1 Hz (1 s) or multiples of either 4 s or 5 s. Process monitoring is usually mapped onto this same 4 Hz (and slower) structure.

Applications with slightly higher loop rates, such as compressor surge control loops running at 12 Hz, are supportable by scheduling multiple communications opportunities per 250 ms cycle.

9.1.9.1.4 Timeslot duration, timeslot alignment, and idle periods

Within 250 ms alignment intervals, time is divided into timeslots of configurable but equal duration. A timeslot is a time interval of predefined duration used to send or receive a Data DPDU and any corresponding ACK/NAK DPDU(s). Timeslots are generally shared by at least one pair of DLEs that communicate during the allocated time. The DL organizes these timeslots into superframes, which are collections of timeslots with a common period and potentially other common attributes.

Timeslot durations are configured during D-subnet setup. Normally, during a given period of operation, all timeslots within a D-subnet have the same duration. Timeslot duration is configured in units of 2^{-20} s (~0,95 μs).

NOTE 1 The DL binds timeslot duration to superframes, and nothing in the standard prevents multiple superframes with different timeslot durations from being active simultaneously within a D-subnet. However, this standard considers only configurations wherein a single timeslot duration is used in a given D-subnet. A DLE supporting multiple timeslot durations simultaneously, such as a backbone router or a bridge between two D-subnets, is modelable as containing multiple DLEs running in parallel.

Timeslots align with the 250 ms alignment intervals, but timeslot durations do not necessarily divide evenly into 250 ms. Therefore, the system inserts a short idle period every 250 ms as needed, thus realigning the timeslots to a 4 Hz cycle. This is shown in Figure 77 with two illustrative examples using different timeslot durations of 9,999 ms and 11,719 ms, respectively.

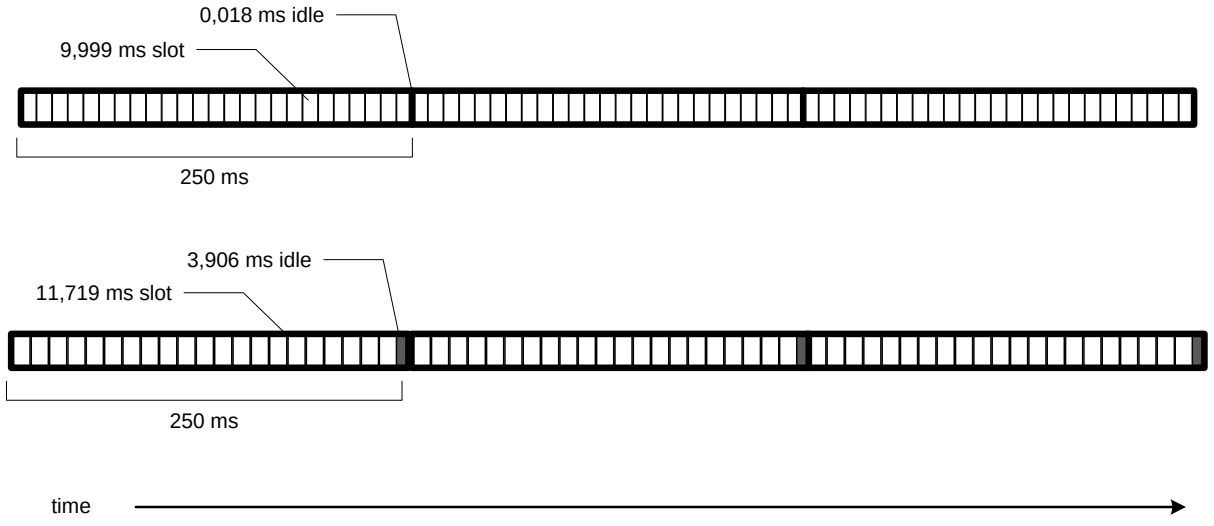


Figure 77 – Timeslot durations and timing

NOTE 2 Calculations for the idle times in Figure 77:

a) $10\,485 \times 25 \times 2^{-20}$ s = 249,982 ms. Subtract from 250 ms to get 0,018 ms;

b) $12\,288 \times 21 \times 2^{-20}$ s = 246,094 ms. Subtract from 250 ms to get 3,906 ms.

NOTE 3 A timeslot intended closely to approximate 10 ms is 10 485 units, or 9,999 275 ms, in duration.

The idle period is not used by the system for scheduled operations; it is simply a short period inserted to ensure that timeslots align and repeat at 250 ms intervals. This makes it straightforward for the system manager to organize collections of timeslots that repeat at exact multiples of 250 ms.

Slow-hopping periods can span an alignment interval. In such cases, slow-hopping periods are not interrupted by idle periods; that is, if a slow-hopping period traverses the edge of an alignment interval, the receiver's radio continues operation through the idle period.

9.1.9.1.5 Scheduled timeslot time

Each DL timeslot has a scheduled fractional-second start time, which is called the scheduled timeslot time. Both the DL and higher-layer protocols use this coordinated time sense as security material, thus providing replay protection. See 7.3.2.6.

Since the 250 ms intervals align with TAI quarter-seconds, scheduled timeslot timing can be derived from the timeslot duration. For example, if timeslot duration is 9,999 ms, scheduled timeslot times in the second quarter-second of TAI time t are t + 0,250 000 s, t + 0,259 999 s, t + 0,269 998 s, etc.

NOTE This mixed-radix time representation facilitates inter-conversion at higher layers, where PDUs often span D-subnets with different data rates, different timeslot durations and/or different DL protocols.

The scheduled timeslot start time is in units of 2^{-20} s (microsecond precision). DLEs commonly use internal clocks based on a 2^{15} Hz (32 KiHz) very-precise very-low-power "watch" crystal. Implementations may, and commonly will, round the scheduled timeslot start time to the nearest 32 KiHz clock tick. This rounding is permitted on a per-timeslot basis, without adjusting the underlying timeslot schedule based on units of 2^{-20} s. However, this rounding to 32 KiHz is not permitted to accumulate over a 250 ms period, as shown in Table 101. This consideration is included within the ±96 μs jitter allowed by this standard; see 9.4.3.3.1.

Table 101 – Approximating nominal timing with 32 KiHz clock

Nominal timeslot offset (ms)	**2^{-20} s (clock counts)**	**2^{-15} s (clock counts, rounded)**	**Actual timeslot offset** (ms)
0	0	0	0
10	10 485 × 1	328	10,010
20	10 485 × 2	655	19,989
30	10 485 × 3	983	29,999
•••	•••	•••	•••
200	10 485 × 20	6 553	199,982
210	10 485 × 21	6 881	209,991
220	10 485 × 22	7 208	219,971
230	10 485 × 23	7 536	229,980
240	10 485 × 24	7 864	239,990

9.1.9.2 DL time propagation

9.1.9.2.1 General

The DLE uses TAI time for its internal operation, and provides a notion of TAI time as a service (through the DMAP) to wireless neighbors compliant with this standard.

In a D-subnet, DLEs may take on three functions in the time propagation process:

- DL clock recipient, a receiver of periodic clock updates through the DL; or
- DL clock source, a provider of periodic clock updates to DL neighbors; or
- DL clock repeater, a DL clock recipient that also acts as a DL clock source to some of its neighbors.

Additional information about time propagation can be found in 6.3.10.

9.1.9.2.2 DLE clock stability

The clock stability of the DLE is the nominal clock stability of a wireless device, in the presence of periodic time corrections from the D-subnet.

The DLE, when configured as a clock recipient, relies on the D-subnet to provide periodic time updates wirelessly. It may also use these time updates in calibration of its internal clock to account for conditions such as temperature, aging, and voltage.

The time reference for a D-subnet originates from one or more clock masters, which provide time that is monotonically increasing at a rate that closely tracks real time.

When the DLE joins the D-subnet, the DLE reports its clock stability to the system manager as dlmo.DeviceCapability.ClockStability (called ClockStability here), which is in units of parts per million (1×10^{-6}). ClockStability applies to any arbitrary 30 s period during the expected life of the device, under all conditions that a user might reasonably expect from the device's published specifications.

For example, if ClockStability reports that a DLE has a clock with maximum instability of 10×10^{-6} s/s, then the DLE's clock shall be stable to within ± 300 μs during any arbitrarily selected 30 s period.

ClockStability is reported as an envelope. For example, if ClockStability reports that a DLE has a maximum clock instability of 10×10^{-6} s/s, then the clock shall be stable to within ± 300 μs at all instants during any arbitrarily selected 30 s period. This is intended to allow for devices that are subjected to occasional environmental shocks that can cause small clock discontinuities. Small discontinuities are acceptable, as long as they do not add up to more than ±ClockStability × 30 μs at any time over any 30 s period.

NOTE 1 ClockStability, specified in units of 1×10^{-6} s/s, is equivalently 1 μs/s.

NOTE 2 It is possible that neighboring DLEs have clocks that drift in opposite directions. Therefore, in the worst case, ClockStability is additive between pairs of neighboring DLEs. In practice clock repeaters are corrected periodically, which lessens that effect.

The standard assumes that clock drift is negligible within a timeslot.

ClockStability is reported to the system manager without caveats. If the device is specified to work under environmental stress, such as extremes of temperature or mechanical shock, then ClockStability shall reflect performance under such stress.

The attribute dlmo.ClockExpire, configured by the system manager, provides the maximum number of seconds that the DLE can safely operate in the absence of a clock update. Normally, the system manager arranges that a DLE will maintain clock synchronization as a by-product of normal communication. However, when the DLE fails to receive a clock update for an extended period of time, defined by ClockExpire, the DLE should actively interrogate a DL clock source for a time update. Failure to do so will eventually result in loss of time synchronization with the D-subnet. ClockExpire defaults to a value that is appropriate for use during the joining process.

A clock repeater should not send clock corrections to its neighbors through ACK/NAK DPDUs if it has not itself received a clock correction for a period that exceeds the ClockExpire attribute. If a clock repeater's clock has expired and it is polled for a time update, it should respond with a NAK1.

A DLE with an expired clock should not be used as a clock repeater, but it may continue to operate in the D-subnet, albeit with a potential risk of losing synchronization with its neighbors. The attribute dlmo.ClockTimeout provides the maximum amount of time that a DLE may reasonably continue operating in a D-subnet in the absence of a clock update. If the DLE has not received a clock update for a period of time that exceeds DLTimeout, the DLE may reasonably reset itself to the provisioned state and initiate a search for a new D-subnet.

NOTE 3 A DLE operating in a slow-channel-hopping configuration is capable of being configured to retain a D-subnet connection for extended periods of time, even with a clock that has drifted across timeslots.

9.1.9.2.3 Preferred and secondary clock sources

The system manager configures each DL clock recipient to treat one or several of its neighbors as DL clock sources. Such DL clock sources may be designated as preferred or secondary, based on the attribute dlmo.Neighbor[ ].ClockSource. Multiple neighbors may be designated as preferred DL clock sources. A DL clock recipient should adjust its clock value whenever it has an interaction with a preferred DL clock source.

The attribute dlmo.ClockStale defines a period of time that shall pass before a DLE begins accepting clock updates from secondary DL clock sources. If, after a period of ClockStale, no clock update is received from any preferred source, the DL clock recipient should accept clock updates in its interactions with neighbors that are designated as secondary DL clock sources. Once a DL clock recipient accepts a clock update from a secondary DL clock source, it should continue to use that secondary DL clock source until either (a) it receives a clock update from a preferred DL clock source, or (b) the secondary DL clock source times out.

The attribute dlmo.ClockStale determines the timeout interval. For example, if ClockStale is set to 45 s by the system manager, then a DL clock source should not accept clock updates

from a secondary DL clock source until it has not received a clock update from any preferred DL clock source for at least 45 s.

Each DL clock recipient can be configured to retain and periodically report statistics (see 9.4.3.9) for any of its preferred DL clock sources, including:

- A count of clock timeout events.
- A running average of clock corrections, a signed integer in units of 2^{-20} s, indicating a bias if nonzero.
- The standard deviation of clock corrections, estimated in units of 2^{-20} s, for example, a value that roughly accounts for approximately 68 % of clock corrections.
- A count of clock corrections in excess of three such standard deviations.

9.1.9.2.4 Shared time sense during D-subnet operation

D-subnet transactions nominally occur in a DL timeslot, on a schedule known to both sender and receiver. This shared sense of time is used as security material, both for header compression and for replay protection.

DL clock sources may be configured to periodically transmit DL advertisements embedded in a Data DPDU's DAUX subheader. Each such advertisement provides a TAI time reference for the D-subnet. All standard-compliant DLEs that receive an advertisement are assumed to be capable of participating in time-synchronized communication with the advertising DLE. DL clock recipients with relatively imprecise clocks may have a limited capability to communicate only with routers that use slow-channel-hopping.

MAC layer authentication requires shared security keys, shared time sense, and knowledge of a neighbor's EUI64Address. This information is acquired stepwise during the joining process, with security keys provided at the end. The DL uses the standard block cipher (usually AES-128), together with a well-known security key, during the joining process in order to provide enhanced integrity checking (but not security). This is described in more detail in 7.4.

In slotted-channel-hopping, both transaction initiators and transaction recipients share an intrinsic sense of which timeslot is being used; otherwise, the DLEs would not be able to communicate. Every timeslot has a scheduled start time that is known by all participating DLEs.

In slow-channel-hopping, a DLE with an inaccurate sense of DL time will nominally transmit in a particular timeslot, based on its own sense of time. However, such a DLE is permitted to miss its target, and may actually transmit in any timeslot within the slow-channel-hopping period. Therefore, unicast Data DPDUs that are transmitted using a slow-channel-hopping superframe shall include an extra octet in the Data DPDU's header to indicate which timeslot within the slow-channel-hopping period was intended. This allows the receiving DLE to reconcile its timeslot sense with that of the transmitting DLE, applying time correction across timeslots to validate the accuracy of the transaction initiator's clock, and to use the timeslot's scheduled start time as security material.

For security purposes, a timeslot's scheduled start time (which is not the start time of a resulting Data DPDU) is passed to the DSC for use in DL-related security operations. See 9.1.11. In most cases, the timeslot of the Data DPDU and the timeslot of any responding ACK/NAK DPDU are the same. There is one exception, which occurs for slow-channel-hopping when the clock of one of the DLEs has drifted into a different timeslot. In that case, the channel-hopping-offset of the acknowledging DLE shall be included in the ACK/NAK DPDU's DMXHR (Table 117) to identify unambiguously the exact time that is to be used for security operations, even if the acknowledging DLE is not acting as a DL clock source for the ACK/NAK DPDU recipient. Absence (i.e., non-inclusion) of the channel-hopping-offset field in the ACK/NAK DPDU implies that the DLE originating the Data DPDU and the DLE originating the ACK/NAK DPDU are using the same timeslot.

9.1.9.3 Pairwise time synchronization

9.1.9.3.1 General

Clock updates are propagated through the D-subnet in the course of normal communications within a timeslot or slow-channel-hopping period. These building blocks are leveraged by the system manager to arrange the propagation of D-subnet time.

DL clock sources use three general mechanisms to propagate clock updates to their neighbors.

- A DL clock source originates a Data DPDU that includes an advertisement. The time is conveyed based on when the Data DPDU is transmitted.
- A DL clock source acknowledges a received Data DPDU in a timeslot. The time is conveyed by measuring when the DL clock source detects the start of the Data DPDU and echoing the result of this measurement in the ACK/NAK DPDU(s).
- A DL clock source acknowledges a Data DPDU within a slow-channel-hopping period. The process is similar to acknowledgment within a timeslot, with the addition of an octet to identify the timeslot uniquely if needed.

The standard relies on stable clocks, particularly in field routers, for reliable D-subnet operation. For DL clock stability requirements, see Table B.8.

A DL clock recipient maintains a list of valid neighboring DL clock sources. If it receives a clock update from a designated DL clock source, it uses the data to update its clock.

When a DLE discovers a D-subnet, it acquires the D-subnet's TAI time and uses that reference for subsequent communication. The DLE shall then periodically re-synchronize its internal clock to the D-subnet clock. These re-synchronization operations are incremental, based on offsets into a timeslot with a scheduled time reference known to both DL clock source and DL clock recipient.

Whenever a DLE interacts with one of its designated DL clock sources, it receives updated clock information. Clock adjustments are included in ACK/NAK DPDUs, thus conveying time information to the recipient. Clock updates are also included in the DAUX subheader of advertisement Data DPDUs originating from DL clock sources.

Thus, a DLE may receive clock updates as a by-product of routing data through a DL clock source. Alternatively, if a DLE needs more frequent clock updates, it may be configured to receive clock updates by enabling its receiver to operate at times coinciding with periodic scheduled Data DPDUs from DL clock sources that include the DAUX subheader with an advertisement.

A DLE may acquire a clock update by sending a Data DPDU with a zero-length DSDU to a DL clock source. A DL clock source shall acknowledge such a Data DPDU and then discard it.

9.1.9.3.2 Clock source acknowledges receipt of a Data DPDU within a timeslot

When a DLE transmits a unicast Data DPDU (see Figure 81) to a DL clock source, it may receive a clock update in the ACK/NAK DPDU, as shown in Figure 78.

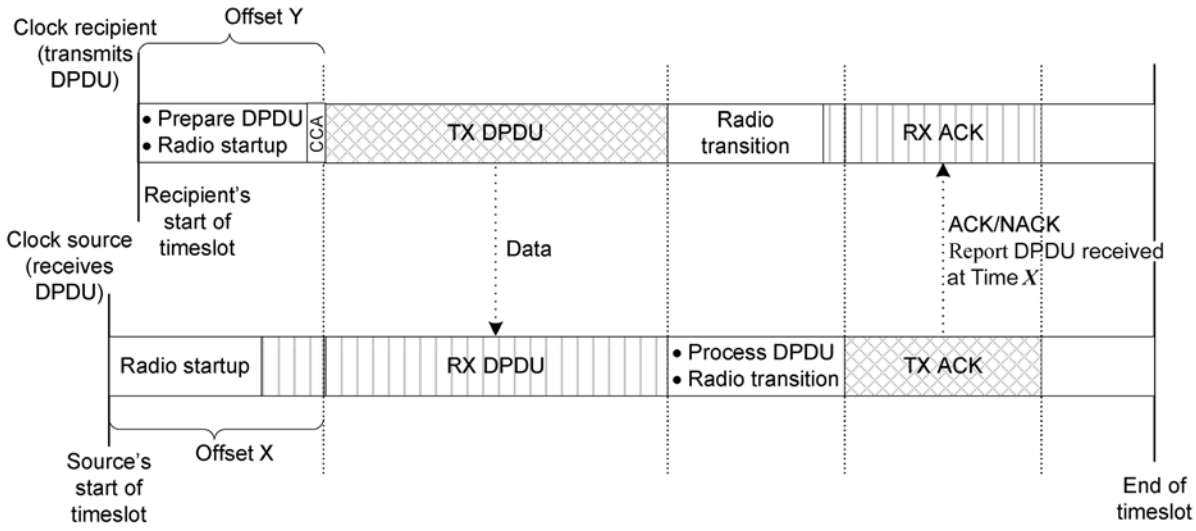


Figure 78 – Clock source acknowledges receipt of a Data DPDU

The ACK/NAK DPDU from the DL clock source includes the start time of the original transmission reported as a time offset (with microsecond precision, in units of 2^{-20} s) of the Data DPDU's start time (i.e., the start time of the PhSDU) from the scheduled timeslot start time as measured by the DL clock source.

A given unicast D-transaction occurs on a single channel, with the Data DPDU and ACK/NAK DPDU(s) all transmitted on the same channel.

This clock synchronization information is targeted at the DLE that originated the Data DPDU. While other DLEs may overhear and update their clocks based on the same information, the design is not optimized for that usage.

While a DLE's internal details of clock synchronization are not specified by this standard, the transaction is intended to support a clock synchronization process that operates generally as follows:

a) The DL clock recipient sends a Data DPDU to the DL clock source and records that it sent the Data DPDU at time offset Y.

b) The DL clock source receives the Data DPDU at virtually the same instant and records that it received the Data DPDU at time offset X.

c) The diagram shows the case where the DL clock recipient's clock is running faster than the DL clock source. In that case, $X > Y$. If the DL clock source is running faster, $X < Y$. The time of the DL clock source is assumed to be correct.

d) In the ACK/NAK DPDU, the DL clock source reports that it received the Data DPDU at time offset X.

e) The DL clock recipient applies a time correction that is computed as $(Y - X)$.

The result shown in the Figure 78 is that the timeslots start at different times but end at the same time. An actual implementation may delay the application of time corrections, such as by adjusting the clock gradually. See 9.1.9.2.2.

9.1.9.3.3 Clock source originates a Data DPDU that includes an advertisement

A DL clock source is often the originator of a Data DPDU. This is generally the case when a DL clock source transmits a scheduled advertisement that includes the TAI time. The payload of the Data DPDU may be unrelated to time synchronization; the TAI time information is

contained within the DAUX subheader, so that it may be overheard by any of the DL clock source's neighbors.

Multiple DLEs may be configured to enable their receivers simultaneously in anticipation of a scheduled Data DPDU conveying an advertisement.

This standard requires that a DL clock source be capable of precisely controlling when advertisement Data DPDUs are transmitted, with a precision of ± 96 μs (3 octets), referenced to the DLE's internal clock. See 9.4.3.3.1.

9.1.9.3.4 Clock source acknowledges a Data DPDU within a slow-channel-hopping period

As noted previously, DL clock sources use ACK/NAK DPDUs to report time with microsecond resolution (2^{-20} s), as offsets relative to the scheduled nominal start time of a timeslot. Identification of the timeslot is assumed to be unambiguous, known to both sender and recipients. Thus only the offset is communicated in clock updates.

However, within slow-channel-hopping periods, a DL clock recipient's internal clock may have drifted tens or hundreds of milliseconds or more relative to the sender's clock, so that the presumption of shared identification of the timeslot might be incorrect. Therefore, an extra octet, identifying a specific timeslot, is added to both Data and ACK/NAK DPDU headers within slow-channel-hopping-periods, as specified in the DPDU's DHDR. The initial timeslot within a slow-channel-hopping period is defined as having an offset of zero.

Within slow-channel-hopping-periods, DLEs with an accurate sense of time should operate within timeslot boundaries. However, DLEs without an accurate sense of time might not be capable of respecting timeslot boundaries, and might not even know which timeslot they are actually using. A timeslot offset in each DPDU header allows the receiver to reconstruct the sender's time sense and vice versa (see 9.3.3.3).

9.1.9.3.5 Auditing the quality of a neighbor's clock

When a DLE joins the D-subnet, it reports its clock stability specifications to the system management function. In their normal interactions with these DLEs, DL clock sources and DL clock recipients can be configured by the system to collect diagnostics, on a per-neighbor basis, to audit these nominal specifications (see 9.4.3.9).

9.1.9.3.6 Discontinuous clock adjustments

Under some conditions, it may be necessary for the system manager to adjust all of its DLE's clocks by tens or hundreds of milliseconds or more, for example, when two D-subnets are being joined. The system manager may schedule a discontinuous clock adjustment to occur at a particular TAI time, by setting the dlmo.TaiAdjust attribute on each of its DLEs. At the designated time, all DLEs shall adjust their clocks forward or backward by the specified amount of time.

There are some system impacts of a clock adjustment that should be considered whenever this feature is used.

- The security model in this standard does not allow time to run backward. Time is used in the security nonce, which can never be repeated in any standard communication layer. Consequently, there shall be an interruption in service, equal to the magnitude of the adjustment plus at least one timeslot, if time is adjusted to an earlier time.
- Phased reports will shift in time by the amount of the clock adjustment, unless corresponding contracts are revised in tandem with the clock adjustment.
- DLEs that do not receive the time correction before the cutover time may lose synchronization with D-subnet operation, and may need to re-discover their neighbors.

9.1.9.4 Transactions within timeslot templates

9.1.9.4.1 General

NOTE Transactions are also described in 4.6.11.

Transaction timing within a DL timeslot is specified by timeslot templates. This standard defines default timeslot templates that are needed by the DLE in its interactions with a wireless provisioning DLE. Additional timeslot templates may be added by the system manager during or following the joining process. (See 9.4.3.3 for more information on timeslot templates.)

Figure 79 illustrates some aspects that are addressed by DL timeslot templates.

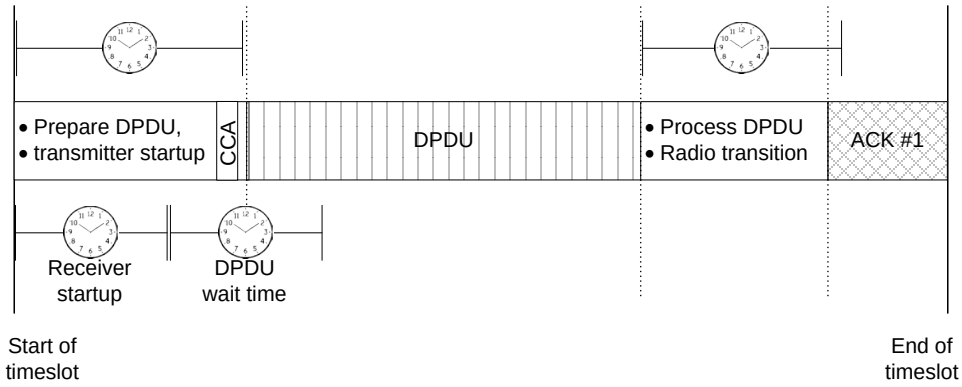


Figure 79 – Transaction timing attributes

As shown in Figure 79, timeslot templates define the timing for operations such as Data DPDU reception wait time, and the turnaround time (Data DPDU reception processing and radio transition) between receiving a Data DPDU and transmitting an ACK/NAK DPDU.

Generally, there are two types of transaction templates: transmit and receive. A transaction initiator template specifies a time range to begin Data DPDU transmission, to check the channel for activity before transmission, and the relative placement of any ACK/NAK DPDU(s) in the timeslot. A transaction receiver template specifies the interval during which a received Data DPDU can begin arriving, and thus when to timeout if no Data DPDU start is detected. It also specifies the timing of any ACK/NAK DPDUs sent by transaction responders.

Default timeslot templates cover baseline transactions needed to interact with a provisioning DLE, which are usable during the joining process. Default templates may also be used by joined DLEs for general transactions. Additional templates may be provisioned into the DLE to accommodate features such as:

- carrier sense multiple access with collision avoidance (CSMA/CA) periods at the start of a timeslot (see 9.1.9.4.8); and
- extended timing of ACK/NAK DPDUs relative to the end of the Data DPDU or the end of the timeslot (see 9.1.9.4.6).

By convention in this standard, timeslot template timing is specified based on the start and end times of both Data and ACK/NAK DPDUs and the end of the timeslot. PhPDU timing, dependent on the details of the physical layer that conveys each DPDU, can be inferred from those DPDU start and end times (see 9.4.3.3).

9.1.9.4.2 D-transaction overview

As shown in Figure 80, the DL supports both unicast and broadcast transactions.

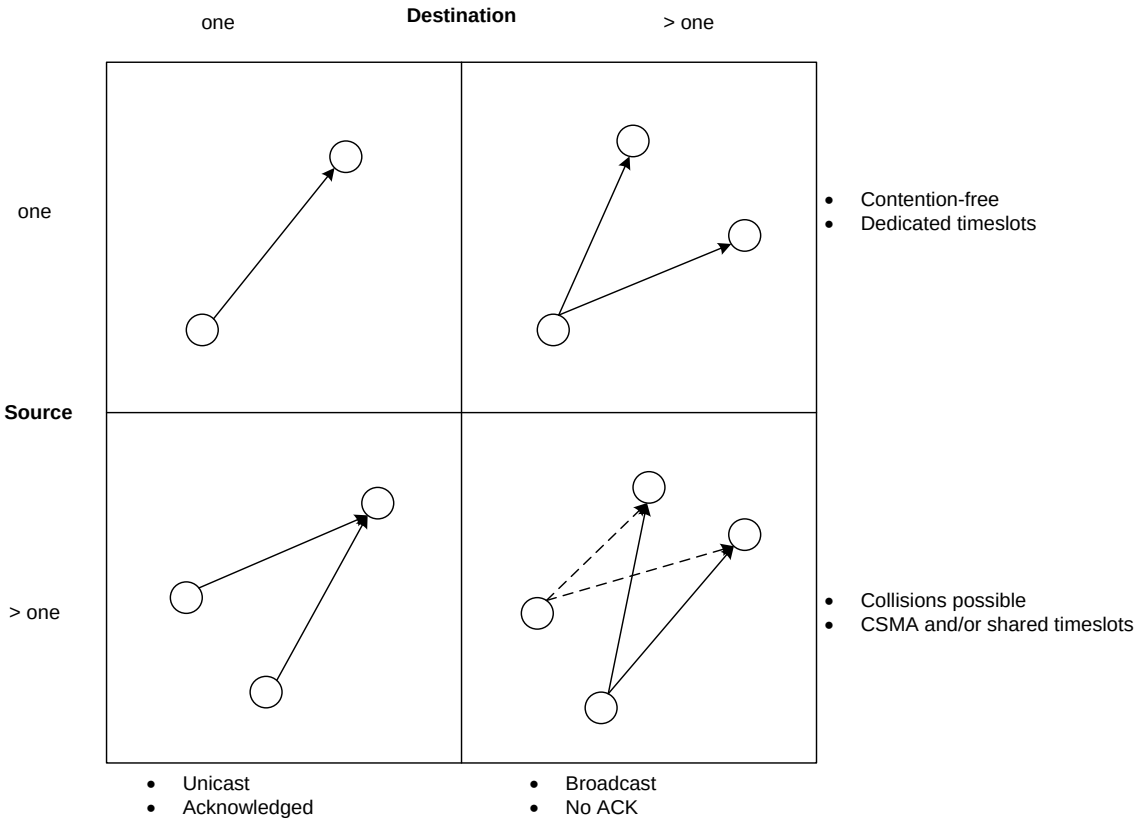


Figure 80 – Dedicated and shared transaction timeslots

Unicast transactions, indicated in the left half of Figure 80, may use dedicated timeslots, for example, when a DLE reports repetitively on a schedule. Duocast is a variant of unicast, wherein a second receiver is scheduled to overhear the Data DPDU and provides a second ACK/NAK DPDU. Duocast is shown graphically in Figure 84.

Receipt of an ACK (positive acknowledgment) DPDU by the transaction initiator indicates that the transaction recipient has successfully received the Data DPDU and that the transaction initiator should mark the transaction as complete. Unicast and duocast transactions require the transmission of ACK/NAK DPDUs in response to such receipt.

Broadcast transactions, indicated in the right half of Figure 80, may also use dedicated timeslots, such as for scheduled advertisements. Broadcast transactions cannot use a DL immediate acknowledgment (i.e., via an ACK/NAK DPDU).

As shown in the lower left quadrant of Figure 80, unicast and duocast transactions may use shared timeslots, such as within slow-hopping periods. Shared timeslots are commonly used for retries, join requests, exception reporting, and burst traffic.

As shown in the lower right quadrant of Figure 80, broadcast transactions such as solicitations may also use shared timeslots.

The term contention-free in Figure 80 is relevant only within the scope of D-subnet timeslot allocation. If two D-subnets, or DLEs within the same D-subnet, are allocating timeslots in an uncoordinated fashion, then access is not contention-free. Likewise, other users of the 2,4 GHz spectrum, such as WiFi, Bluetooth®[7], ZigBee, IEC 62591 and IEC 62601, potentially

[7] Bluetooth® is the trade name of a product supplied by the Bluetooth Special Interest Group. This information is given for the convenience of users of this standard and does not constitute an endorsement by IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

may also interfere. Improved coexistence with these uncoordinated systems is achieved by using CCA (see 9.1.9.4.3) to check the channel before transmission.

The use of broadcast in this standard is limited to these DL operations: advertisements and solicitations. Advertisements and solicitations, used for D-subnet discovery, are described in 9.1.13.

NOTE Broadcast as described here does not use broadcast MPDUs defined in IEEE 802.15.4. See 9.1.5. Broadcast and multicast, as AL services, are not supported in this version of the standard.

9.1.9.4.3 Unicast transaction

Figure 81 illustrates a unicast transaction.

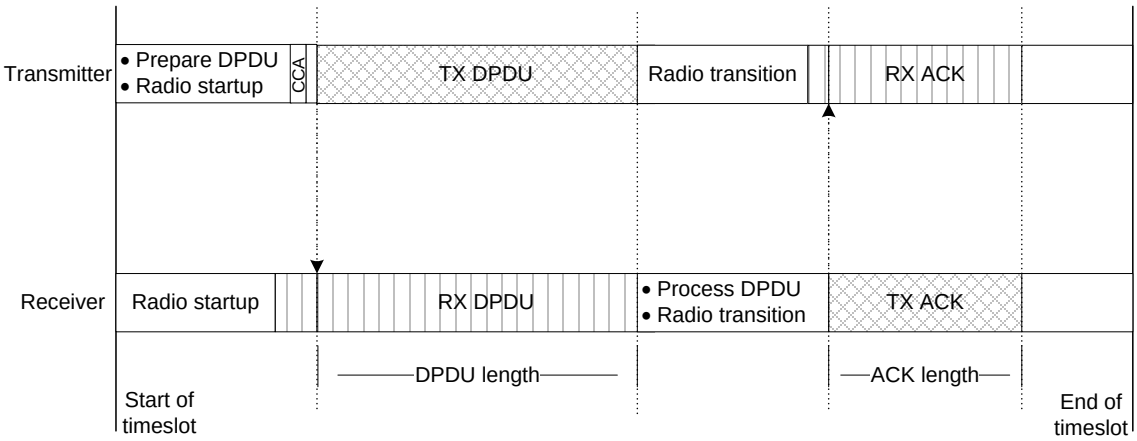


Figure 81 – Unicast transaction

Before a DLE transmits a Data DPDU in a timeslot, it prepares the Data DPDU for transmission, generally with DL security. Prior to transmission, the DLE is normally configured to perform a clear channel assessment (CCA) of the radio space.

CCA shall be implemented as described in IEEE 802.15.4. That standard specifies a detection period of 8 symbols. CCA shall be performed as configured in the timeslot template field dlmo.TsTemplate[ ].CCAmode (see Table 163), where the choices, listed by their coding, are:

- CCA Mode 4 (Aloha);
- CCA Mode 1 (energy above threshold);
- CCA Mode 2 (carrier sense only);
- CCA Mode 3 (carrier sense and/or energy above threshold).

Compliance to IEEE 802.15.4 requires that at least one of the CCA modes be supported. Compliance to ETSI EN 300 328 requires that CCA Mode 1 be supported. If a non-zero value for dlmo.TsTemplate[ ].CCAmode is selected and the selected mode is not supported by the DLE, the DLE may choose a different CCA mode that is supported by the DLE and permitted by the applicable regulatory regime. CCA modes supported by the DLE are indicated in dlmo.DeviceCapability.SupportedCCAmodes; see 9.4.2.23.

IEEE 802.15.4 permits CCA Mode 3 to be implemented either as the AND or the OR of CCA Mode 1 and CCA Mode 2. When CCA Mode 3 is implemented as the AND of CCA Mode 1 and CCA Mode 2, it may be used in lieu of CCA Mode 1 in regulatory regimes that require CCA Mode 1. When CCA Mode 3 is implemented as the OR of CCA Mode 1 and CCA Mode 2, it cannot be used in lieu of CCA Mode 1 in regulatory regimes that require CCA Mode 1.

If a specific CCA mode is required by regulation, but is not supported by implementation, then the DLE shall not initiate transactions.

If CCA reports a busy medium, the transmission transaction shall be aborted.

Use of CCA as defined by IEEE is not intended to exclude additional CCA checks that might be supported by advanced devices. For example, if a backbone router is capable of detecting IEEE 802.11 modulation, the DLE may reasonably leverage this capability to detect and report a busy medium.

CCA should be complete 192 μs prior to the start of the physical layer header.

If the D-transaction requires an ACK/NAK DPDU, the transmitting DLE enables its receiver after the transmission is completed, at a time specified in the TsTemplate. If an ACK DPDU is received, the transmitted Data DPDU is deleted from the DLE's transmit queue.

When a DLE is scheduled to receive a Data DPDU, it enables its PhLE's receiver at the time specified in the TsTemplate and waits for the expected PhPDU. If it detects a valid IEEE 802.15.4 SHR and PHR, it continues to attempt to receive the entire PhPDU. It then processes the contained Data DPDU (including DMIC authentication) and determines if the Data DPDU requires an acknowledgment. To send an ACK/NAK DPDU in acknowledgment, a DLE that is a transaction responder enables its PhLE's transmitter and sends the ACK/NAK DPDU within the same timeslot, such that the ACK/NAK DPDU is transmitted at the time specified by the timeslot template for the primary (or a secondary, etc.) responder in that timeslot, depending on the DLE's role as primary, etc., responder.

The time window for each expected ACK/NAK DPDU is defined by the timeslot template. If there is a substantial delay between the end of the Data DPDU and the scheduled start of the expected ACK/NAK DPDU, an implementation may sensibly power down its receiver during the delay.

9.1.9.4.4 Negative acknowledgments

A transaction recipient shall respond to receipt of a unicast Data DPDU with a NAK DPDU when it cannot accept the Data DPDU at that time, but has successfully received it without other error. Time synchronization information may be included in NAK DPDUs. Similarly, a NAK DPDU ensures that RF statistics correctly log a clean transmission. A NAK DPDU can be used to exert back-pressure as a simple flow control mechanism.

The DL supports two types of NAK DPDUs: NAK0 and NAK1. A NAK0 DPDU is intended to indicate resource limitations in the router, while a NAK1 DPDU is intended to signal downstream connectivity problems in the D-subnet.

The DLE shall respond to a unicast Data DPDU with a NAK0 DPDU when it correctly receives the Data DPDU but cannot accept it due to lack of capacity in its message queue. See 9.1.8.5 for a discussion of the DLE's message queue. A DLE may also respond with a NAK0 when it is configured in excess of its forwarding capability (ForwardRate), as described in 9.4.2.23.

The DLE may respond to a unicast Data DPDU with a NAK1 DPDU to apply back pressure in the event of lost downstream connectivity. For example, when the DLE loses downstream connectivity to all of its next neighbors in a specific graph, and then receives a Data DPDU that is following the same graph, the DLE may sensibly generate an immediate response of a NAK1 DPDU to indicate the lack of ability to forward Data DPDUs that are directed to the same graph.

When a DLE receives a NAK0 or a NAK1 DPDU from a neighbor, it shall back off by not transmitting more Data DPDUs to that neighbor for a period of dlmo.NackBackoffDur. This backoff delay does not include delay of Data DPDUs without a payload, which allows the DLE

to poll a neighbor that is a DL clock source for a time update even though the neighbor is not accepting Data DPDUs at that time.

As described in 9.1.9.2.2, if a DL clock repeater's clock has expired and it is polled for a time update, it should respond with a NAK1.

9.1.9.4.5 Explicit congestion notification

This standard supports explicit congestion notification (ECN) as described in IETF RFC 3168. ECN provides a mechanism for a router to affect AL flow control.

As described in 12.12.4.2.3, there are a limited number of data source requests that can be simultaneously awaiting response from a data sink. Flow control at the data source operates by incrementally increasing the limit on outstanding requests, based on receipt of timely data sink acknowledgments.

ECN provides a mechanism whereby flow control can be effective without driving the D-subnet to the point of failure. Any router along the path from data source to data sink may set ECN to indicate that the router is nearing its capacity. When the data sink receives the ECN, it echoes it back to the data source, which then accounts for the ECN in its flow control logic.

An ECN indicator in the Data DPDU header may be set by any field router that is experiencing congestion, following the guidelines in IETF RFC 3168, as a signal to a data source that it should apply flow control to reduce its use of D-subnet resources. This ECN indicator is propagated to the data sink, such as a gateway, and eventually works its way back to the data source through a TL acknowledgment.

In addition, the DL provides a special type of acknowledgment called ACK/ECN. A DLE receiving an ACK/ECN treats it as a normal DL acknowledgment. In addition, the DLE may treat the ACK/ECN as an early indication that the data sink's acknowledgment will include an ECN.

Use of the ACK/ECN should be limited to Data DPDUs with a priority of seven (7) or less, corresponding to best effort queued and real time sequential flows.

9.1.9.4.6 Data DPDU wait times

The clock times of transmitting and receiving DLEs are rarely in perfect synchronization. Therefore, a transmitting DLE is unlikely to transmit a PDU (protocol data unit) at exactly the time that a receiving DLE expects it. If a PDU is transmitted too early, the receiver might not yet be enabled. If a PDU is transmitted too late, the receiving DLE may have disabled its radio in order to save energy. PDU wait time (PWT) is the time period when the receiver is expected to listen for incoming PDUs. A particular degree of timing accuracy between the DLEs is implicit in the system manager's selection of PWT.

DLEs compliant with this standard accommodate configurable PWTs and configurable timeslot durations. PWT is not directly specified in this standard, but can be inferred within timeslot templates from the earliest and latest times that PDU reception begins. See Table 161.

NOTE 1 This tutorial does not make a distinction between PhPDU and DPDU timings. As specified in 9.4.3.3, the DL follows a convention of specifying timeslot timing in reference to the DPDU (PhSDU), and not to the PhPDU. In implementations based on IEEE 802.15.4 (2,4 GHz), PhPDU header detection involves receipt of a preamble, start frame delimiter (SFD), and frame length, for a total PhPDU header of 192 μs (6 octets) before the DPDU begins. Similarly, radio transmission of a PhPDU begins 192 μs prior to the nominal DPDU start time.

PDU wait times in a unicast PDU are illustrated in Figure 82. The same principles apply to other types of PDUs.

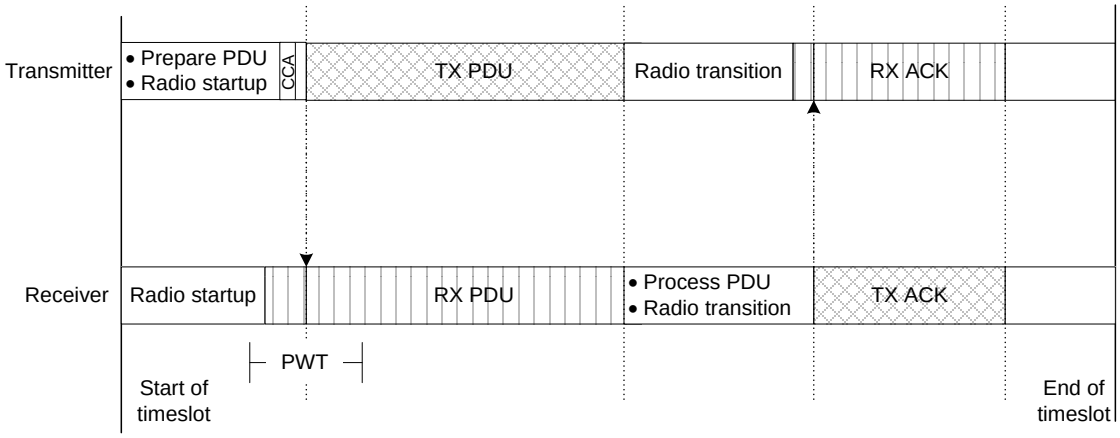


Figure 82 – PDU wait time (PWT)

The duration of the PWT is configured by the system manager accounting for the intrinsic stability of transmitting and receiving DLE's clocks, and the guaranteed maximum time between clock updates.

For example, if transmitting DLEs are accurate to ±1 ms relative to the receiver within the clock expiration period (dlmo.ClockExpire), the receiver's PWT should be 2 ms long. A less accurate transmitting DLE will require a longer receiver PWT or a shorter clock expiration period. (In this example, the radio's listening time should be slightly longer than the 2 ms PWT, to account for the PhPDU's SHR and PHR durations.)

If the receiver does not begin receiving the expected PDU by the end of its PWT, the receiver is permitted to disable its radio for the duration of the timeslot.

Timeslot durations of 10 ms have a timing budget that can allocate about 2 ms to PWT. For a longer PWT, either other allocations have to be adjusted or the timeslot duration has to be increased. For example, if the D-subnet supports DLEs that sleep for up to two minutes, timing errors of about ±2 ms may accumulate between reports. This can be accommodated by increasing the PWT from (for example) 2 ms to 4 ms, with a corresponding increase in timeslot duration and receiver energy consumption.

NOTE 2 DLEs with infrequent reporting intervals are capable of being configured to check the D-subnet periodically for receivable Data DPDUs. These DLEs are capable of receiving clock corrections through the DAUX subheader as part of the same process.

9.1.9.4.7 Duocast/N-cast transactions

Duocast/N-cast is a variant of unicast, wherein one or more additional receivers are scheduled to overhear the Data DPDU and provide additional acknowledgments. Duocast/N-cast provides support for latency-controlled access to a backbone with an increased probability of first-try success. Duocast/ N-cast transactions are intended primarily for DLEs with links to two or more infrastructure DLEs, particularly to backbone routers, as shown in Figure 83.

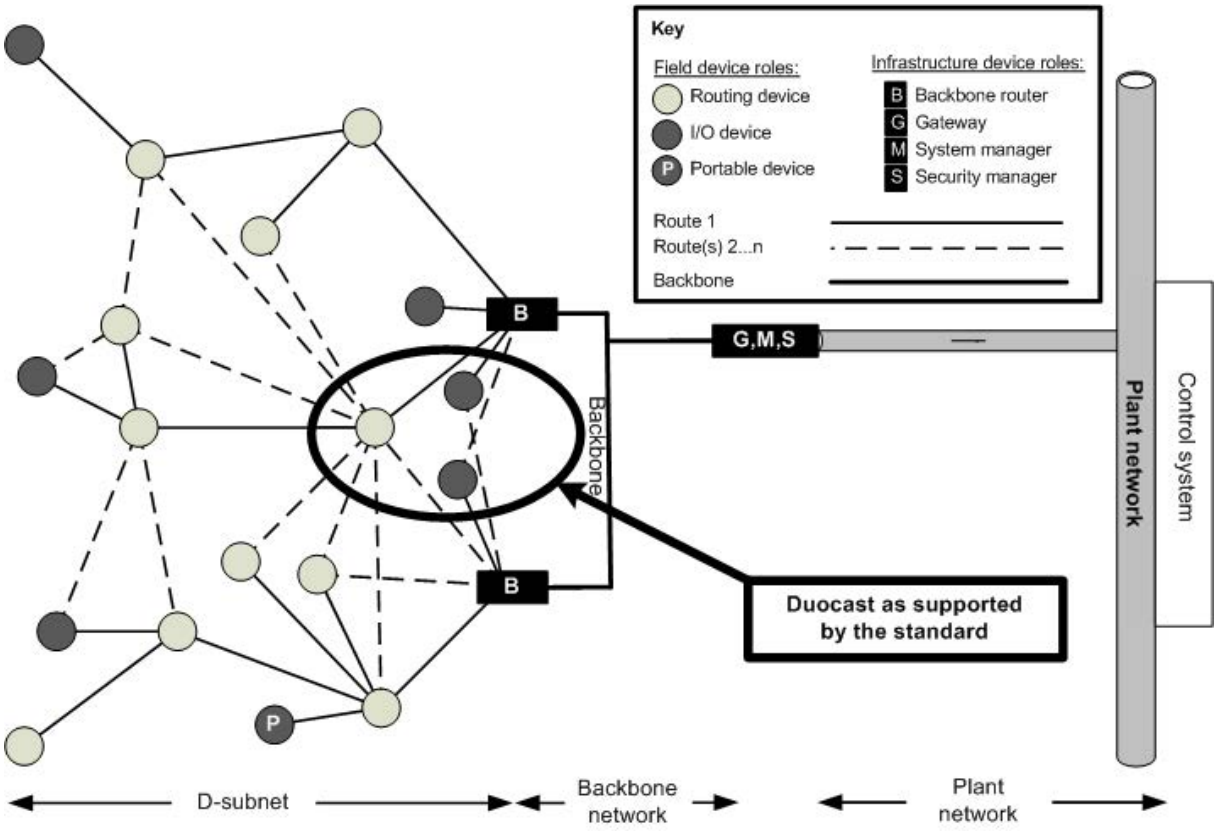


Figure 83 – Duocast support in the standard

DLEs receiving the duocast/N-cast acknowledgments, circled in Figure 83, need to be configured with timeslot templates with an extended listening window for the additional acknowledgment(s). DLEs sending the secondary duocast/N-cast acknowledgments (e.g., one of those gray DLEs circled in Figure 83) need to be configured individually with timeslot templates with appropriately delayed/deconflicted acknowledgment windows for their individual acknowledgments, and with the address of the primary intended recipient to enable them to respond only to the expected Data DPDU. Duocast/N-cast support usually involves increased timeslot duration of approximately 1 ms to 2 ms per secondary receiver, as configured by the system manager.

NOTE 1 Coordination of the duocast/N-cast response may involve back-channel coordination between the responding infrastructure DLEs (since such responders usually are backbone routers that are also connected to a higher-throughput backbone), but coordination via standard configuration messaging is also possible, depending on the design of the infrastructure DLEs.

NOTE 2 If the probability of success of a single acknowledged-unicast transaction is p, the probability of failure for such a unicast transaction is $(1\text{-}p)$. For the corresponding duocast transaction it is typically $(1\text{-}p)^2$, while in the general case for an N-cast transaction it is $(1\text{-}p)^N$. Thus when p = 95 %, the probability of failure for a unicast transaction is 5 %, for a duocast transaction it is 0,25 %, for a 3-cast transaction it is 0,0125 %, etc. Thus, in most relatively planar environments duocast suffices, though in highly metallic and obstructed three-dimensional environments such as offshore oil platforms 3-cast ("triocast") may be worth consideration.

NOTE 3 In many cases duocast is for contracts with a small maximum APDU size, such that the transaction-initiating Data DPDU and both acknowledging ACK/NAK DPDU subslots would require no greater duration than a maximal-payload unicast transaction with a single-acknowledgment Data DPDU and its subsequent single acknowledging ACK/NAK DPDU. In that case a single timeslot duration is usable for both unicast and restricted duocast transactions, albeit with different transaction templates.

Duocast/N-cast timeslots may be scheduled in conjunction with available digital bandwidth for a fast retry on an alternate channel, as shown in Figure 66.

Figure 84 illustrates a transaction involving duocast transmission and reception.

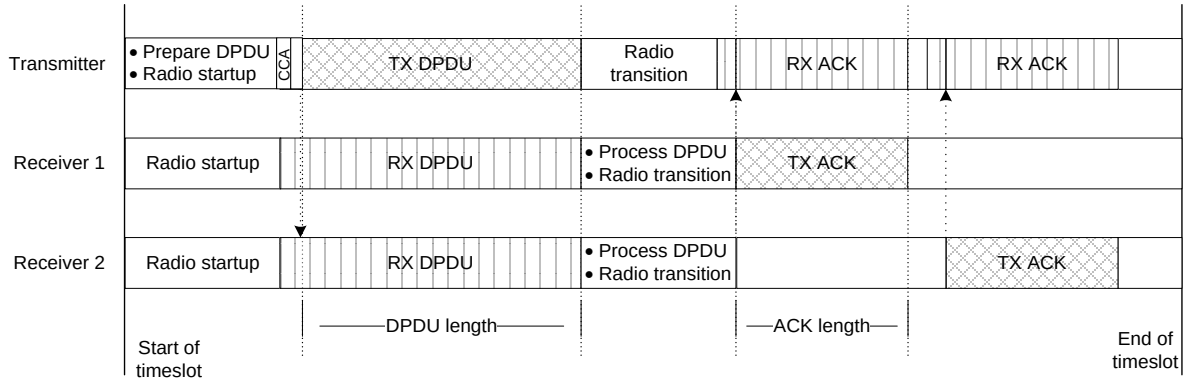


Figure 84 – Duocast transaction

In the example in Figure 84, the Data DPDU is addressed to receiver 1, the primary recipient, and is overheard by receiver 2, a secondary recipient. For duocast/N-cast transactions, the destination address of the Data DPDU is set to that of the primary recipient (receiver 1 in Figure 84), and an acknowledgment is expected from at least one recipient during the same timeslot. As illustrated in Figure 84, the primary recipient transmits an ACK/NAK DPDU upon receipt of the Data DPDU. Secondary recipients also transmit an ACK/NAK DPDU, but after an additional recipient-dependent delay to allow time for any preceding ACKNAK DPDUs to complete.

If an ACK DPDU is received from any acknowledging recipient, the transaction is complete and the Data DPDU is deleted from the transaction-originator's DLE's message queue.

If an ACK DPDU is received from a recipient, the DLE that originated the transaction is not required to expend energy receiving and processing any subsequent ACK/NAK DPDU(s) in that transaction. However, the DLE that originated the transaction should periodically verify that it is able to receive an ACK/NAK DPDU from each expected acknowledging recipient, to confirm the continuing availability of the secondary receiver(s).

As noted in 9.1.5 and described in 9.3.4, a duocast/N-cast acknowledgment from a secondary recipient includes the acknowledging DLE's own address field in the source address field of the MHR. This enables the transaction initiator to identify the acknowledgment's source and occasionally to report the same to the system/security manager to facilitate detection of cyber-attacks.

9.1.9.4.8 Shared timeslots with CSMA/CA

Unicast transactions may occur in timeslots that are dedicated to a specific link. Alternatively, shared timeslots may be designated to provide bandwidth on demand to a collection of DLEs. Shared timeslots are usually configured to transmit only near the start of the timeslot.

Timeslot templates specify transmission time as a range as per 9.4.3.3. At a minimum, the transmit time shall be configured to a range of at least 192 μs, thereby allowing for ± 96 μs (± 6 PHY symbol periods) of jitter that is permissible in a transaction initiator. If the transmission time range is configured to be larger than 200 μs, the DLE shall select a randomized time within that range to begin its transmission, making reasonable accommodation for the DLE's actual transmission jitter characteristics.

Figure 85 illustrates the use of a shared timeslot with active CSMA/CA.

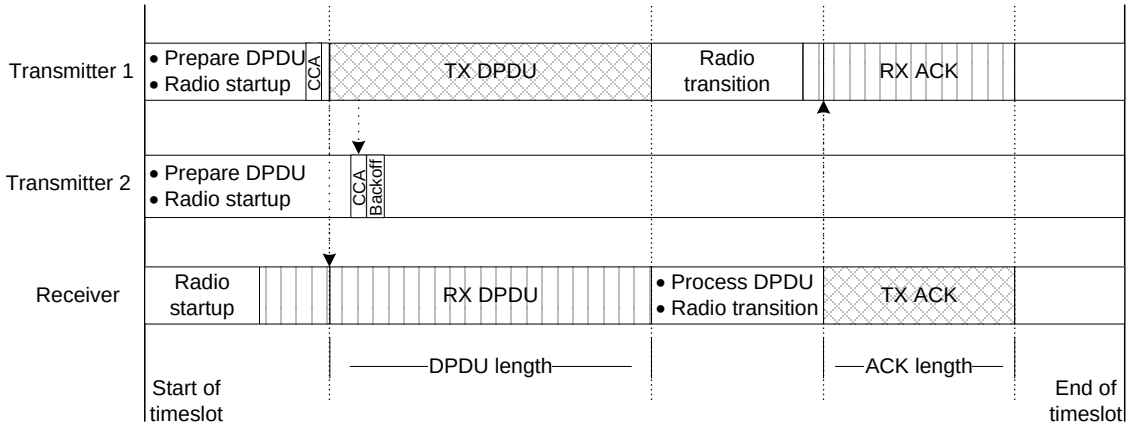


Figure 85 – Shared timeslots with active CSMA/CA

In the example in Figure 85, two DLEs are contending for use of the channel in a shared timeslot. This approach is used for all shared slots, configurable to be used in various ways.

Priorities within a shared timeslot may be managed by configuring DLEs with different timeslot templates. A DLE with a high-priority Data DPDU, such as a retry for a failed duocast transaction, may be configured to transmit its Data DPDU as early as possible within the timeslot. A DLE with less critical requirements may be configured to delay its transmission to slightly later in the timeslot, such as 2 ms later. If another DLE has already claimed the timeslot, as shown in Figure 85, the CCA of the delayed DLE might (or might not) detect that the channel is in use and consequently defer its transmission to another timeslot.

Use of active CSMA/CA within shared timeslots may involve configurations with longer timeslots and longer Data DPDU wait times, and use of more receiver energy.

9.1.9.4.9 Transactions during slow-channel-hopping periods

Some DLEs do not have a sufficiently stable time base to communicate at their normal messaging rate within short timeslots. These DLEs may need to use slow-hopping periods for their communication. Slow-channel-hopping periods are simply a set of concatenated timeslots on the same channel, wherein the receiver runs its radio continuously and the transaction initiator is not required to respect timeslot boundaries within the slow-hopping period.

As shown in Figure 86, a transaction during a slow-channel-hopping period is very similar to a unicast transaction in a shared timeslot, except that Data DPDU transmission can occur anywhere within the slow-channel-hopping period. Transmitting DLEs target the beginning of a specific timeslot within a slow-channel-hopping period, based on the transaction initiator's own sense of time, which is not required to be very well synchronized with that of the intended receiver(s). Transmitting DLEs use CCA to check the channel before transmission. In the absence of higher priority operations (such as forwarding of Data DPDUs), a receiver hosting the slow-channel-hopping period runs its radio receiver continuously except when responding to Data DPDUs that it receives.

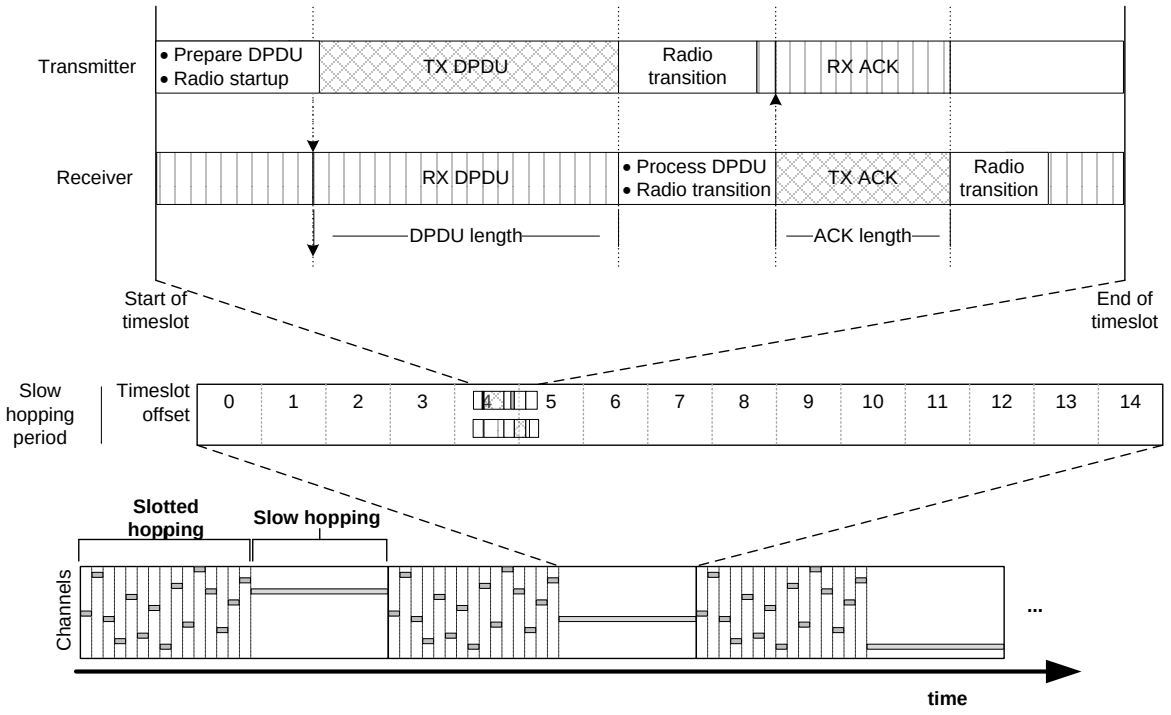


Figure 86 – Transaction during slow-channel-hopping periods

DLEs can target any timeslot within a slow-channel-hopping period, with one major caveat: scheduled Data DPDU timeslot time is required to increase with each transaction.

DLEs should respect timeslot boundaries within slow-channel-hopping periods to the best of their ability. If all DLEs within slow-channel-hopping periods are well behaved, with well-synchronized clocks, the resulting performance is approximately comparable to that of the slotted Aloha protocol.

A DLE whose time sense differs from that of its intended receiver(s) will nominally transmit near the start of a particular timeslot, based on its own sense of time. However, such a DLE may actually initiate transmission in any phase of any timeslot within the slow-channel-hopping period.

For example, a DLE that has a clock source with a stability of $\pm 50 \times 10^{-6}$ over a few-minute interval, due to uncompensated environmental fluctuations, may sleep for 3 min between transactions, corresponding to a clock drift of about ± 9 ms. A DLE of this type might wait for a scheduled advertisement to get itself resynchronized prior to transmission. Alternatively, it might transmit during a slow-channel-hopping period (if available) and receive time synchronization in the acknowledgment. Continuing with this example, a DLE with ± 9 ms accuracy would select one of the slots within the slow-channel-hopping period, and transmit using the appropriate link template. The DLE would nominally be transmitting in a particular timeslot, based on the DLE's own sense of time, but may actually be transmitting in a different timeslot.

In this example, the DLE should not attempt to transmit in the first or last timeslot in the slow-channel-hopping period, because it may actually transmit outside of the available time range. It is the DLE's responsibility to avoid selecting timeslots that are inconsistent with the DLE's time-keeping capabilities, accounting for the DLE's own uncalibrated clock drift in combination with the 10×10^{-6} clock drift allowed for a neighboring router or the 100×10^{-6} clock drift allowed for a neighboring non-routing field device.

All Data DPDUs in slow-channel-hopping periods shall include an extra octet in the Data DPDU header to indicate which timeslot offset within the slow-channel-hopping period was intended. This enables the receiving DLE to reconstruct timeslot information from the transmitting DLE, to apply time correction across timeslots, to validate the accuracy of the transaction initiator's clock, and to use the scheduled timeslot start time as security material for message authentication. (This is specified in the Data DPDU MAC subheaders, DMXHR; see 9.3.3.4.)

If necessary, a corrected timeslot offset is provided in the acknowledgment (see 9.3.4). Unambiguous shared timeslot identification is needed for both the transaction initiator and transaction receivers to authenticate the Data DPDU and to resynchronize time. Even if the transmitting DLE has a highly accurate sense of time, the receiving DLE(s) might not; therefore the channel-hopping-offset octet is required for all Data DPDUs using a slow-channel-hopping superframe.

The initial timeslot within a slow-channel-hopping period is defined as having an offset of zero.

9.1.10 D-subnet addressing

9.1.10.1 Address types

DL16Addresses shall always be used within a D-subnet, except that EUI64Addresses are used in a limited way during the initial phases of the joining process to communicate with the joining device.

Every DLE in a D-subnet is identified in three ways:

- Each D-subnet DLE has an EUI64Address identifier that is presumed to be unique.
- Each DLE compliant with this standard shall be assigned at least one IPv6Address when it joins the D-subnet. However, within the DL, only the DL16Address alias for this IPv6Address (described next) shall be used.
- Each DLE or foreign device that is accessible through a D-subnet has a D-subnet-unique DL16Address, which is an alias for its IPv6Address. The scope of any DL16Address shall be limited to a particular D-subnet.

The EUI64Address shall be used as a new DLE's address for immediate neighbor addressing prior to and during the joining process. Once a DLE has joined the D-subnet and received its IPv6Address, it shall be addressed by either its IPv6Address or a D-subnet-local DL16Address alias of that IPv6Address.

The EUI64Address is also used in each DPDU security nonce. Whenever a DL16Address is used in a Data DPDU or an ACK/NAK DPDU, the DPDU's recipient(s) need(s) a-priori knowledge of the corresponding EUI64Address. This neighbor information is provided by the system manager as part of the link establishment process. An exception is made for a new DLE that is communicating with a neighbor that has advertised its DL16Address. In that case, the DLE of the joining DLE shall acquire the EUI64Address of the advertising DLE by initiating a transaction with that neighbor, sending it a Data DPDU with a null payload while requesting the EUI64Address in the replying ACK/NAK DPDU, as described in 9.3.3.3. For that bootstrapping transaction, the applicable D-key is K_global, used with a DMIC-32, which is the same as for other Data DPDUs involving an EUI64Address. (See 9.1.11 for discussion of DL security.)

When a DL16Address refers to a DLE within a D-subnet, the DL16Address is always the 16-bit MAC address of the DLE. When a Data DPDU contains a DL16Address that refers to a device not within the scope of the D-subnet, the Data DPDU is routed to a DLE that is serving as a backbone router, which maps the logical DL16Address to its IPv6Address counterpart.

Most Data DPDUs include two source DL16Addresses and two destination DL16Addresses. One pair of source/destination DL16Address is for next-hop addressing at the MAC sublayer. A second pair of source/destination DL16Addresses (actually D-aliases) are found within the Data DPDU's DADDR subheader, where they specify the D-subnet-local ultimate source and destination NLEs via their D-aliases.

Routing is usually specified by graphs, not by D-addresses. However, when source routing is used within a D-subnet, DL16Addresses are normally used. Again, the one exception is that EUI64Addresses are used during the joining process for communication with an immediate neighbor within the D-subnet.

9.1.10.2 Subnet identifier and uniqueness of DL16Addresses

The scope of a DL16Address is a D-subnet. A neighboring D-subnet may use the same DL16Address to reference a different DLE. Different D-subnets may use different DL16Addresses to refer to the same backbone device.

Each D-subnet has a 16-bit D-subnet identifier (dlmo.SubnetID; see 9.4.2.1), which has the same value as the PAN ID found in the IEEE 802.15.4 MPDU header. Within the scope of a network, each D-subnet shall have a unique dlmo.SubnetID. However, different networks are not necessarily coordinated, so dlmo.SubnetIDs across co-located standard-compliant networks are not guaranteed to be unique. Thus, it is possible, though unlikely, that a DPDU from a different standard-compliant D-subnet will be received with what appears to be a valid DL16Address and PAN ID. The DLE relies on the DSC to discard such DPDUs on receipt due to DMIC mismatch or DMIC non-authentication, where that mismatch occurs due to non-identical D-security symmetric keys.

SubnetID=0x0000 and SubnetID=0xFFFF shall not be used as D-subnet IDs in this standard. SubnetID=0x0001 is reserved for provisioning D-subnets (see 13.1) and shall not otherwise be used.

9.1.11 DL management service

9.1.11.1 General

Management messages to a DLE are fully secured at the AL, using the end-to-end relationship between the system/security manager and the DLE's DMAP.

The DL's MAC, based on IEEE 802.15.4, is not accessed directly by the DMAP; instead it is configured indirectly through the DMSAP. This isolates the rest of this standard from evolutionary changes to IEEE specifications, facilitates future adoption of alternative physical layer specifications (e.g., radios) that may have alternative or enhanced associated MACs, and enables some MAC and PHY operational aspects (such as CCA) to be used or not on a timeslot-by-timeslot basis.

As shown in Figure 87, DL management commands generally flow through the full communication protocol stack defined by this standard.

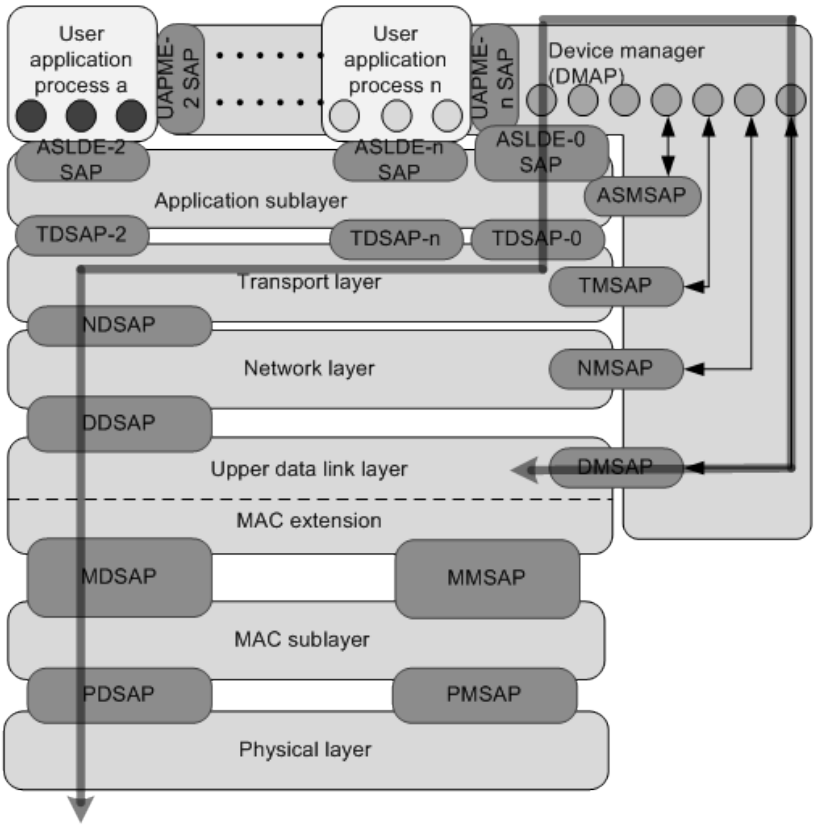


Figure 87 – DL management SAP flow through standard protocol suite

A DLE is configured via its DMAP through its DMSAP. Most management SAPs are generic, simply reading and setting data structures within the layer. Those data structures generally define how a DLE operates its state machine. The DMAP communicates with the system manager through the application sublayer, using end-to-end security.

For information on the general handling of standard management objects, see 6.2.5 and 6.2.6.

9.1.11.2 Management attributes and indexed attributes

DMSAPs involve the manipulation of the DL management object (DLMO). The DLMO includes a variety of attributes that are used to configure the DLE and/or report its status.

Some DLMO attributes apply to specific values. For example, attribute dlmo.SubnetID provides the subnet-ID for the D-subnet that the DLE has joined.

Some DLMO attributes can be visualized as tables with a collection of indexed rows. Each such attribute is specified as an indexed OctetString. For example, the DLE includes the attribute dlmo.Neighbor, which is an indexed OctetString attribute containing a collection of neighboring DLEs. Each entry of the dlmo.Neighbor attribute includes a set of fields for that neighbor, such as an indicator of whether that neighbor is a DL clock source. Each entry of the dlmo.Neighbor table is uniquely identified by the neighbor's DL16Address. Indexed OctetString attributes in the DLMO include:

- dlmo.Ch: channel-hopping patterns;
- dlmo.TsTemplate: timeslot templates;
- dlmo.Neighbor: DLE neighbors;
- dlmo.NeighborDiag: diagnostics for DLE neighbors;

- dlmo.Graph: graphs for routing;
- dlmo.Superframe: superframes specifying common attributes for associated links;
- dlmo.Link: links, each of which is associated with a superframe;
- dlmo.Route: routes usable in DROUT subheaders.

Relationships among these attributes are described in 9.4.3.1.

DLMO attributes shall be maintained through the DMAP using methods that are described in Clause 9 and Clause 12. The ASLE services Read and Write described in Table 271 provide a general framework for reading and writing DLMO attributes. Methods for writing attributes that are to become active at a TAI cutover time, as well as methods for reading and writing indexed OctetString attributes, are described in 9.5.

The format of the DLMO attributes is defined in the DLE specification. When these objects are embedded within over-the-air messages, the specified formats shall be used. This standard does not (and cannot) constrain how data is stored within a DLE, or define how corresponding information is relayed internal to a DLE.

9.1.11.3 Management messages from immediate neighbors

Any Data DPDU may include a DAUX subheader, for example carrying advertisement information from an immediate neighbor. The DAUX subheader information is not propagated to higher layers of the communication protocol suite. In most cases, the content of the DAUX subheader is intended for the recipients DLE's management process, which in turn may use this information to configure the DLE state machine. For example, the DAUX subheader may include superframe definitions that are intended to be used by the DLE as a starting point in the joining process and/or for neighbor discovery.

The DAUX subheader is modeled as being instantly visible to the DMAP and immediately acted on if necessary.

NOTE Since standards apply only to externally-visible aspects, the internal DMSAP interface is not subject to standardization.

9.1.11.4 Multiple D-subnets

DL management objects support only one active D-subnet at a time. All DL16Addresses shall be unique within the scope of that single D-subnet. This constraint does not prevent a DLE from participating in multiple D-subnets simultaneously. Multiple D-subnets might be modeled as multiple instances of a DLE, but such operation is not specified by this standard.

If dlmo.SubnetID=0, the DLE is not yet participating in a D-subnet.

9.1.11.5 Multiple PhLEs (radios)

Each DSAP of a DLE is assumed to be associated with only one PhLE (radio). This does not preclude implementations with multiple radios, which might be modeled as multiple instances of a DLE. Such operation is not specified by this standard.

9.1.12 Relationship between DLE and DSC

The relationship between the DLE and the DSC is described in 7.3.2. Careful review of 7.3.2 is essential for anyone who wishes to fully understand DLE operation.

Generally, the DLE relies on the DSC for authentication, integrity and conditional confidentiality. All DPDUs include a DMIC that unambiguously validates that an MPDU originates from a DLE that shared the same D-subnet key, vs. those that implement another protocol using similar modulation, coding and channels. The DMIC uses a non-secret security

key during the joining process but then is usually configured to use a shared-secret key once the DLE is joined.

ACK/NAK DPDUs are expanded by the DLE, by inserting the Data DPDU's DMIC into the ACK/NAK DPDU's header, as a virtual field before being processed by the DSC. This virtual field, which is not transmitted, is included in the ACK/NAK DPDU's DMIC calculation. Thus the Data DPDU's DMIC is essentially echoed in the ACK/NAK DPDU without actually transmitting the field. In this manner, the ACK/NAK DPDU is unambiguously bound to the corresponding Data DPDU of the D-transaction.

9.1.13 DLE neighbor discovery

9.1.13.1 General

Wireless D-subnets compliant with this standard are detected by the DLE. A new DLE may hear advertisements from neighboring DLEs that have already joined a D-subnet of interest. This is called neighbor discovery. Following neighbor discovery, the DLE can join the D-subnet as described in 7.4.

The DL advertisement and neighbor discovery processes are the building blocks that enable a DLE to learn the DL16Address and EUI64Address of a neighboring router, and the set of scheduled links that have been allocated for use when joining. That information is then used by the DLE and DME to join the D-subnet.

DLEs discover D-subnets through advertisement Data DPDUs received from one or more advertising DLEs that periodically announce their presence. An advertising DLE is capable of acting as a proxy in the provisioning or joining process. An advertisement contains information that enables a new DLE to send a join request to the advertising DLE, for relay to a system manager, and some time later to receive a join response.

After joining a D-subnet, the DLE receives advertisement Data DPDUs from neighboring DLEs in the D-subnet and builds a local list of candidate neighbors with which it may have reasonable quality communications. This list of candidates is reported to a system manager through the attribute dlmo.Candidates. The system manager uses this information to determine how the DLE fits into the D-subnet topology. In turn that information is used to establish communication relationships between the new DLE and its neighbors.

Advertising DLEs may be discovered using passive scanning, active scanning, or a combination of passive and active scanning.

In passive scanning, the DLE periodically listens for advertisements on a series of channels. Generally, a battery-powered passive scanning DLE will listen frequently when first powered on. If a D-subnet is not discovered quickly, the DLE may extend its battery life by scanning less frequently and/or by allocating shorter time intervals for each scan. Such reductions can result in substantial delays in D-subnet joining and/or D-subnet formation, so are used only after the rapid join strategy has failed.

Active scanning overcomes some disadvantages of passive scanning. DLEs that are configured for active scanning will search for a D-subnet by periodically transmitting solicitation Data DPDUs, which trigger advertisement Data DPDUs from neighboring routers in response. The DLE transmitting the solicitation Data DPDU is called an active scanning interrogator, while the responding DLE is called an active scanning host. Active scanning hosts expend energy operating their radio receivers while listening for solicitation Data DPDUs. Some active scanning hosts have energy available for continuous receiver operation. Active scanning hosts with more limited energy sources may be configured to listen continuously for certain periods of time, such as during D-subnet formation.

Link schedules used for joining are not necessarily related to superframe schedules used for normal D-subnet operation. Thus, little information about D-subnet operation needs to be conveyed in the advertisements.

9.1.13.2 Auxiliary subheader and advertisements

DL advertisements and solicitations are conveyed in a Data DPDU's auxiliary subheader (DAUX) within the DPDU header (DHR). See 9.3.1 for an overview of the DHR. A Data DPDU that conveys a solicitation is known as a solicitation DPDU. Similarly, a Data DPDU that conveys an advertisement is known as an advertisement DPDU.

The DAUX subheader is usually absent from a DHR, but shall be included in any DHR if so configured for a particular link. A Data DPDU containing a DAUX subheader may also carry a higher layer payload that is unrelated to the neighbor discovery function.

Transmission of an advertisement is triggered by an advertisement flag as configured within a link definition. The advertisement flag indicates that an advertisement shall be transmitted in a superframe's timeslot, in the absence of a higher priority link.

The same link definition may also include a transmission flag, in which case the DLE shall check the message queue for matching outbound Data DPDUs. Thus, the Data DPDU may simultaneously carry:

- an advertisement; and
- a DSDU payload that is entirely unrelated to the advertisement.

The payload capacity of the Data DPDU is reduced when an advertisement is embedded within the DHR. Some Data DPDUs on the message queue, particularly messages that have been fragmented at the NL, may be too long to be combined with an advertisement. Such messages are not candidates for links that are shared with an advertisement, effectively giving the advertisement priority access to those timeslots.

NOTE When messages are fragmented by the NL, the fragment size is set by the NL before being passed to the DLE, with fragment size being configured by the system manager. In configurations where advertisements are infrequently combined with transmit links, the maximum fragment size often is limited by the Data DPDU payload capacity without considering the combined advertisement.

In general, Data DPDUs containing an advertisement use a DMIC based on the D-subnet's security key, thereby providing an advertisement that can be trusted by DLEs after they have joined the D-subnet. This DMIC cannot be validated prior to joining, because an unjoined DLE does not yet have the D-subnet security key. Therefore, an unjoined DLE that is scanning for a D-subnet is permitted to process an advertisement in a DAUX subheader even if it is unable to authenticate the Data DPDU containing the DAUX.

Without the benefit of a DMIC, an unjoined DLE receiving an advertisement can still use the IEEE 802.15.4 FCS as an integrity check. However, the IEEE 802.15.4 FCS alone does not filter MPDUs from other systems that use IEEE 802.15.4. Therefore this standard provides an additional integrity check, not involving a security key, specifically covering the advertisement subheader, as specified in 9.3.5.2.4.3.

9.1.13.3 Active scanning solicitation and response

In active scanning a new DLE, acting as an active scanning interrogator, periodically solicits advertisements from active scanning hosts that happen to be in radio range. A DLE receiving the solicitation can be configured to respond with an advertisement.

Active scanning is intended for configurations in which an advertising DLE, in its capacity as an active scanning host, is able to operate its receiver more or less continuously in a slow-channel-hopping configuration. Active scanning hosts may be continuously powered.

Alternatively they may be energy-constrained DLEs that run their receivers in a slow-channel-hopping configuration for limited periods of time, such as during D-subnet formation.

The solicitation request is encoded in the DAUX subheader of the solicitation Data DPDU. A solicitation Data DPDU shall not contain an NL payload.

A solicitation Data DPDU, when received by an active scanning host DLE, causes that DLE to transmit an advertisement Data DPDU in the next timeslot if the DLE is so configured. A router receiving a solicitation Data DPDU shall respond by transmitting an advertisement Data DPDU in the next full timeslot if, and only if:

- the default receive link for scanning (Table 165) applies in the next timeslot and occurs on the same radio channel as the solicitation; and
- the DLE attribute dlmo.ActScanHostFract is configured for response to solicitation Data DPDUs. Dlmo.ActScanHostFract indicates the fraction of time that the DLE should respond when it receives an active scanning solicitation, where
  - a value of 0 indicates that the DLE is not configured as an active scanning host and that it will not respond to solicitations,
  - a value of 255 indicates that the DLE should always respond to solicitations, and
  - a value in the range 1..254 indicates that the DLE makes a uniformly-random selection from the range 1..255 each time it receives a solicitation, and does not respond with a solicitation if the result is greater than dlmo.ActScanHostFract, thus generating a solicitation response Data DPDU with probability dlmo.ActScanHostFract / 255; and
- the DLE is configured to respond to the D-subnet ID included in the advertisement Data DPDU, as described in 9.4.2.20. A solicitation Data DPDU may be configured to include a D-subnet ID to limit respondents to a desired set of D-subnets.

The next full timeslot, in this context, shall be defined as the next timeslot that starts following the end of the solicitation's PhPDU plus 1 ms . Within that next timeslot, the advertisement Data DPDU shall be transmitted using timing as defined in the default transaction initiator template in Table 166, even if that default template is overwritten during DLE configuration.

An active scanning interrogator, which is presumably not synchronized with D-subnet timing, should enable its radio to receive an advertisement Data DPDU starting as early as 3 212 μs following the end of the solicitation Data DPDU, which is 1 ms plus the 2 212 μs from Table 166. Following that time, the active scanning interrogator should keep its receiver enabled long enough to receive an advertisement Data DPDU beginning at any time during a full timeslot duration. The active scanning interrogator should use its own timeslot duration as the assumed timeslot duration of the active scanning host. Therefore, when solicitation Data DPDUs are used, the active scanning interrogator should be configured with a timeslot duration that matches or exceeds the timeslot duration of the target D-subnet.

A solicitation Data DPDU is required for a link that is configured as a solicitation link.

To support passive scanning by new DLEs, active scanning hosts may also be configured to transmit advertisement Data DPDUs periodically.

It may be necessary to suppress solicitations, to account for situations where it is unsafe or illegal for a DLE to operate its radio transmitter without authorization. To address such situations, the DLMO attribute dlmo.RadioSilence provides a mechanism to disable solicitations along with all other DLE transmissions.

Solicitations are disabled by default. Solicitations are not used in the default configuration, and dlmo.ActScanHostFract defaults to zero.

9.1.13.4 Continuous scanning

Neighbor discovery should be an ongoing process even after a DLE has joined the D-subnet. Ongoing scans can, over time, help to form a more optimal D-subnet. Mains-powered routers that spend a substantial portion of their time listening can, over time, receive many advertisements from nearby routers. Battery-powered devices cannot spend a high percentage of their time listening, but even if they just sample the channel periodically, they can, over extended periods of time, build a comprehensive picture of neighboring routers. If the D-subnet is configured with a coordinated schedule of advertisements, such scanning can be performed more efficiently.

A system management function may establish an overall D-subnet schedule for advertisements, and a joined DLE may be provisioned with a schedule of receive links to ensure that such advertisements are heard over time. When used, this approach enables DLEs on the D-subnet to find neighbors efficiently. Additionally, a low-duty-cycle DLE may be configured to use these scheduled advertisements to remain time-synchronized with the D-subnet.

Advertisements are authenticated with a DMIC, to enable DLEs that have joined the D-subnet to rely on scheduled advertisements as a trusted source of timing and connectivity information.

9.1.14 Neighbor discovery and joining – DL considerations

9.1.14.1 General

The DL provides a configurable mechanism to discover neighboring DLEs.

During provisioning, a DLE is configured to scan for neighbors that can act as proxies in the joining process. When an advertisement is received from a candidate neighbor, the DLE uses information in the advertisement to create communication links to and from that neighbor, and then provides the neighbor's addressing information to the DMAP. For the remainder of the joining process, the DLE provides communication support to upper layers by passing Data DPDUs to and from the neighbor.

After joining the D-subnet, the DLE is implicitly or explicitly instructed by the system manager to scan for new neighbors for a designated period of time, using superframes and links provided by the system manager. The result of this scan is reported as neighbor information to the system manager that uses the information to provide the DLE with an optimal configuration within the DLE's mesh. The DLE then continues to accumulate information about new candidate neighbors throughout its lifecycle, and this information is periodically reported to the system manager to facilitate configuration of improved and adaptive mesh configurations.

The D-subnet joining process from the DLE's point of view can be informatively summarized as follows:

– The DLE, possibly in its factory state, searches for an advertisement from the provisioning DLE. This search is built into the DLE as defined by this standard.
– The DLE receives an advertisement from the provisioning DLE. This advertisement, with D-subnet ID = 1, provides a compressed but fully functional configuration for the DLE's state machine. The DLE uses this configuration to communicate with the provisioning DLE. During provisioning, a different DLE configuration, that is subsequently used to search for the target D-subnet, is written to the device provisioning object (DPO).
– At the end of provisioning, the provisioning DLE sets Join_Command=1, indicating successful completion of the provisioning process and causing the DLE to reset to the provisioned state. The DLE defines that reset as first resetting the DLE to its factory state, thus erasing the material from the provisioning DLE's advertisement, and then initializing the DLE with the settings stored in the DPO.

– The DLE then commences operation of its state machine, using the configuration as initialized by the DPO. This configuration from the provisioning DLE should be matched to the target D-subnet to facilitate efficient D-subnet discovery. For example, if the target D-subnet is configured to transmit advertisements on three channels, the DPO might reasonably configure the DLE to scan those same three channels.

The discovery process is successful when the DLE receives an advertisement from the target D-subnet. This advertisement contains a compressed but fully functional DLE configuration that can be used for communication with the system manager through the router that transmitted the advertisement.

If the joining process times out, the DLE is reset to the provisioned state. The configuration derived from the advertisement is erased, the DPO configuration is re-established, and the DLE resumes its search for a target D-subnet.

If the joining process is successful, the DLE's joining configuration persists until explicitly updated by the system manager. Thus, at the end of the joining process, the DLE usually retains an interim connection with the system manager through the advertising router.

Following a successful join, the DLE searches for a set of candidate routers that can be used for communication. After a configurable period of time, defaulting to 60 s, the DLE reports this list of candidates to the system manager. This information is used by the system to replace the interim connection with a more permanent and resilient DLE configuration.

If the DLE's intended reporting rate will be so low that the time synchronization required for slotted-channel-hopping will not be maintainable, and if the local D-subnet provides slow-channel-hopping intervals, then the DLE can shift to using slow-channel-hopping for most of its infrequent communications.

NOTE The joining process, which is intended for infrequent, acyclic use by any given DLE, uses slotted-channel-hopping; slow-channel-hopping during the joining process is not supported.

9.1.14.2 DLE states

When a device is manufactured, the DLE is in its default state. Attributes in the default state are defined by this standard.

In the default state, the DLE shall periodically scan for a provisioning D-subnet, using a search procedure defined by this standard. When the DLE receives one or more advertisements from a provisioning DLE in a provisioning D-subnet, the DLE shall use information in one of the advertisements to establish a superframe with links that can be used to communicate with the selected provisioning DLE. The DLE then informs the DPO (device provisioning object) that a D-association has been established, and switches the DLE to the provisioning state.

In the provisioning state, the DLE halts the search procedure defined by this standard. Instead, the DLE operates its state machine according to a superframe with links that were provided by the provisioning DLE's advertisement. During the provisioning process, the DLE provides communication services to upper layers by operating its state machine according to the advertised superframe and links, so that provisioning APDUs can be passed between the DPO and the provisioning DLE via the DLE that is being provisioned.

If the provisioning process times out, the DLE reverts to its default state and resumes its default search procedure for a provisioning D-subnet.

If the provisioning process is successful, the DPO provides the DLE with a set of attributes, including D-subnet information, superframes, and links, that the DLE can use to search for the target D-subnet(s). The DLE then switches from the provisioning state to the provisioned state, and operates its state machine as configured in the superframes and links that were provided by the DPO.

The switch from the provisioning state to the provisioned state is triggered when the provisioning DLE sets Join_Command=1 (Table 10). When that occurs, the DLE is reset to its provisioned state and then operates its state machine as configured in order to search for a target D-subnet.

In general, a DLE reset to a provisioned state is accomplished in two steps. First, the DLE is reset to its default state. Then, information from the DPO's Target_DL_Config attribute is applied to the DLE. This provides a set of attributes, including D-subnet information, superframes, and links, that the DLE uses to search for the target network and corresponding D-subnets. See 13.8.

If the DLE is provisioned through an out-of-band mechanism, the provisioning state may be bypassed and in that case the DLE transitions directly from the unprovisioned state to the provisioned state.

The DPO retains a copy of the information that was used to provision the DLE, providing a means to reset the DLE back to its provisioned state by putting the DLE into its default state and then adding the provisioned attributes from the DPO.

A DLE in the provisioned state operates its state machine as configured in its provisioned superframes and links. The provisioned superframes and links should be matched to the operating characteristics of the target D-subnet(s), so that a target network is efficiently discovered when the DLE and a target D-subnet are in proximity to each other. The result is that the DLE receives at least one advertisement from at least one proxy DLE that is participating in one of those target D-subnets. The DLE uses the information conveyed by one of the received advertisements to establish a superframe with links that can be used to communicate with the sending proxy DLE. The DLE that is attempting to join the D-subnet then informs its DMAP that a D-association has been established to a neighboring proxy, and that DLE then switches from the provisioned state to the joining state.

When the DLE enters the joining state, it retains its configuration from the provisioned state (see 13.8), and adds the superframe, links and other attributes defined or implied by the advertisement. The DLE then provides communication services to upper layers during the joining process, by operating its state machine according to the advertised superframe and links with the result that joining APDUs are passed between the DMAP and proxy DLE through the DLE that is being provisioned.

If the DMAP's joining process times out, the DLE resets to its provisioned state and resumes scanning for a D-subnet using the provisioned superframes and links.

If the DMAP's joining process is successful, the advertised superframes and links continue to be used temporarily for communication with the system manager. The DPO retains the information needed to reset the DLE back to the provisioned state in the event of a DLE reset.

When the DLE is first placed in its joined state, it has a single connection to the D-subnet through the neighboring proxy DLE, using the superframe and links defined in the advertisement. This single connection, selected implicitly when the DLE selected the proxy DLE, lacks path diversity, and may be suboptimal for any number of reasons. Therefore, the system manager shall provide instructions for the DLE to search for several neighbors and to report the result when the search is completed. Simple instructions may be provided through the same advertisement that established the initial connection to the proxy DLE, or more elaborate search instructions may be provided by the system manager immediately after the DLE joins the D-subnet. In either case, the DLE provides the system manager with a list of candidate neighbors soon after it joins the D-subnet, with 60 s being the default reporting time as controlled by the attribute dlmo.DiscoveryAlert. The system manager analyzes this list of candidate neighbors and then provides the DLE with an updated configuration that includes path diversity (mesh) and generally provides a more optimal D-subnet connection.

To support security processing, the DMAP needs the EUI64Address of the advertising DLE during both the provisioning and joining process. Since the advertisement Data DPDU provides only the advertising router's DL16Address, the DLE shall acquire the EUI64Address from the neighbor when communication begins. The neighbor's EUI64Address shall be acquired by using a transmit link to interrogate the neighbor using a Data DPDU with a null payload, setting the DHDR request EUI64Address bit (bit 5, Table 118) to a value of 1, causing the neighbor's EUI64Address to be returned in the ACK/NAK DPDU.

9.1.14.3 Consolidated DL configuration information

The DPO maintains the attribute Target_DL_Config that includes the settings for various attributes in the DLE. This OctetString is provided to the DLE at the end of the provisioning process, and is retained by the DPO in the event that it needs to reset the DLE back to its provisioned state.

The DL_Config_Info OctetString contains a collection of attributes that the DPO provides to the DLE in order to establish the provisioned state. Each attribute is expressed as a tuple including the attribute number, followed by an OctetString that contains the new attribute value. The structure of DL_Config_Info is shown in Table 102.

Table 102 – DL_Config_Info structure

Octets	Bits							
	7	6	5	4	3	2	1	0
1 octet	N (number of attributes)							
1 octet	$AttributeNumber_1$ (Unsigned8)							
–	$NewAttribute_1$ (OctetString)							
…	…							
1 octet	$AttributeNumber_N$ (Unsigned8)							
–	$NewAttribute_N$ (OctetString)							

Several of the attributes in DL_Config_Info are indexed OctetStrings. In these cases, $NewAttribute_X$ is a new row entry. By DL convention, each row entry in an indexed OctetString attribute includes the row index as its first field.

DL_Config_Info can be used to configure any read/write attribute in the DLE. At a minimum, DL_Config_Info shall configure:

- AdvFilter, which provides a filter so that the DLE can select superframes that are of interest.
- At least one superframe, and at least one link, that can be used by the DLE in searching for advertisements.

Timeslot templates used for searching for the D-subnet, if different from the default timeslot templates, shall be provided to the DLE during the provisioning process.

DL_Config_Info shall not be used except through the DPO.

The DLE's provisioned attributes shall be retained by the DPO so that the DLE can be reset to its provisioned state.

Superframe operation may be delayed or disabled by setting the IdleTimer field within the superframe. For example, two superframes may be provisioned in a DLE, with superframe number1 searching aggressively for the D-subnet and superframe number2 searching on a low-duty cycle. The IdleTimer of superframe number1 might cause that superframe to time out

a few minutes after the DLE is configured. Alternatively, a superframe might be configured with an IdleTimer so that it is idle until some future time. If the DLE is reset to the provisioned state, the superframe idle timers shall be reset to the originally provisioned state as well.

A DLE in the provisioned state shall operate its DL clock and increment TAI time. This enables the DLE to operate its superframes as provisioned to discover candidate D-subnets. During the joining process, a revised TAI time will be received in advertisements and the DL clock reset accordingly.

The DLE might completely lose its time sense while in the provisioned state, for example due to removal of a battery. Complete loss of time sense in a provisioned DLE shall trigger a reset of the DLE to its provisioned state.

9.1.14.4 Scanning for neighbors in the unprovisioned state

An unprovisioned DLE begins in the system default state. Its DLE configuration includes the five default channel-hopping patterns and the three default timeslot templates. Its superframes, links, graphs, and routes are blank.

Before attempting to join the plant network, the unprovisioned DLE needs to establish contact with a provisioning DLE and be transitioned to the provisioned state. This may occur through an out of band mechanism, such as a wired modem or an infrared link.

This standard defines an unprovisioned DLE's search procedure for a provisioning D-subnet's advertisement. The search procedure is radio silent, not involving solicitations or any other transmission until an advertisement is received from a provisioning D-subnet.

An unprovisioned DLE shall scan for a provisioning D-subnet's advertisements on channels 4 and 14, corresponding to IEEE 802.15.4 channels 15 and 25, if radio regulations so permit (see Figure 59). In each of these channels, the unprovisioned DLE shall scan for an advertisement at a fixed interval of exactly 0,25 s, 0,5 s, 1 s, 2 s, 4 s, 8 s, 16 s, or 32 s. When the DLE is powered on or physically reset, it shall scan at the shortest interval of 0,25 s for at least 10 s, and then may gradually increase the interval as necessary to preserve its energy supply.

SubnetID=1 is reserved for the provisioning D-subnet. Therefore, only advertisements with SubnetID=1 shall be considered by a DLE in the unprovisioned state.

9.1.14.5 Scanning for neighbors in the provisioned state

Once a DLE is in the provisioned state, it shall use its provisioned superframes and links to scan for a D-subnet of interest. The DLE may discover multiple advertisement routers and then select one to be used as a proxy in the joining process.

9.1.14.6 Scanning for neighbors after joining the D-subnet

A DLE joins the D-subnet through a single neighbor that is sending advertisements. Initial contracts are established based on a route that is not necessarily optimal, through the joining proxy DLE. Having established a single connection through the joining process, the DLE shall be configured by the system manager to immediately search for additional and alternative neighbors in order to support a more optimized mesh configuration.

A DLE in the joined state can be configured by the system manager to passively scan for advertisements by configuring the DLE with links and superframes that enable the DLE's radio receiver at scheduled times. The DLE may be configured to actively scan for advertisements using solicitations.

The NeighborDiscovery alert (see 9.6.2 and 9.4.2.24) provides a mechanism for the DLE to transfer this information to the system manager. The alert is intended for the following scenarios:

- Within 15 s of entering the joined state, the DLE shall be configured with superframes and links that search for advertisements from routers that are in range, and select the most promising candidates. The configuration can be accomplished through the advertisement itself, prior to joining, as described in 9.3.5.2.4.2. Alternatively, the configuration can be provided by configuring DLMO attributes after joining. After a configurable elapsed amount of time, as defined by the attribute dlmo.DiscoveryAlert, the DLE shall use the neighbor discovery alert to send the list of candidates to the system manager. A superframe dedicated to this initial search can be configured to automatically time out through its IdleTimer field.
- The DLE should be configured by the system manager to continuously passively and/or actively scan for advertisements on an ongoing basis. Over time, this enables the DLE to build a list of candidate neighbors. The HRCO may be configured to periodically transmit this candidate neighbor table, along with neighbor diagnostics, to the system manager. The system manager may alternatively read the candidate neighbor table, dlmo.Candidates, on its own schedule.
- A DLE shall also use the neighbor discovery alert whenever a connectivity issue is reported through the DL_Connectivity alert (see 9.6.1). This provides an up-to-date picture of neighborhood connectivity, enabling the system manager immediately to consider alternative solutions to the reported DLE connectivity problem.

9.1.15 Radio link control and quality measurement

9.1.15.1 General

Signal quality information is accumulated in the DLE and reported through the DMAP. In support of these higher-level functions, the DLE provides primitives that report signal quality information. The DLE also provides attributes that enable the system manager to control radio emissions.

9.1.15.2 Performance metrics

Numerous performance metrics can be accumulated by the DLE on a per-neighbor basis, configured by the system manager and reported through the DMAP (see 9.4.3.9).

The DLE can be configured by the system manager to accumulate the following types of performance data on a per-neighbor basis:

- Received signal strength indicator (RSSI) and received signal quality indicator (RSQI).

NOTE This standard defines practices for RSSI and RSQI reporting to facilitate consistency, so that signal characteristics are somewhat comparable among devices of differing construction.

- For transmissions: counts of successful transmissions, CCA backoffs, unicast errors, and NAKs.
- Count of received Data DPDUs.
- Diagnostics of clock corrections.

Per-channel diagnostics are also collected and consolidated for all neighbors.

RSSI shall be reported as a signed 8-bit integer, reflecting an estimate of received signal strength in dBm. RSSI reports shall be biased by +64 dBm to give an effective range of -192 dBm to +63 dBm. For example, a reported RSSI value of -16 corresponds to a received signal strength of -80 dBm.

The actual received signal strength for a device depends on the receiver's noise floor, which is device-construction and operating-temperature dependent, as well as on the receiver's

minimum sensitivity. These can be accounted for within the receiving device, as they are quite repeatable for a given device design and device operating temperature. Thus device designers should approximately map available indications of received signal strength and device temperature (where known) to a reasonable estimate of RSSI, so that system managers have a consistent basis for making routing decisions among DLEs.

RSQI shall be reported as a qualitative assessment of signal quality, with higher number indicating a better signal. A value of 1..63 indicates a poor signal, 64..127 a fair signal, 128..191 a good signal, and 192..255 an excellent signal. A value of zero indicates that the chipset does not support any signal quality diagnostics other than RSSI.

RSSI is a quantitative measurement, mapped to physical units. RSQI is a qualitative measurement. RF devices from different manufacturers, or with different part numbers, or even with an improved die layout or photolithography shrink, may generate different raw RSQI values. Since the IEEE 802.15.4 PHY does not specify a common measurement and reporting methodology for the underlying hardware, no superior software sublayer can create it; the information simply is not present consistently across different devices.

RSQI metrics are intended to be particularly useful when comparing different entries in dlmo.Candidates, where the assessment is among signaling of differing origins received by a single device. A DLE may use innovative techniques to report comparisons of the likely link quality of different candidate neighbors. Because inter-device variances at the receiving device are removed, RSQI entries in dlmo.Candidates reported from a single given DLE may be reasonably compared to each other, and fine distinctions can be taken as meaningful. For example, differences within the range of good signals can be reasonably taken into consideration if the RSQI metrics are from the same DLE.

RSQI may also be compared across different DLEs, but fine distinctions are unlikely to be as meaningful. For example, the distinction between a fair and an excellent link is likely to be meaningful even if reported from unlike devices, but distinctions between different levels of good links has no standard meaning if reported from different devices.

Since any reported RSQI value is a qualitative measurement, comparison of such values shall necessarily take the RF-chip-specific nature of such measurements into account. Given the specified interpretation of reported values, any two non-zero RSQI values reported by different DLEs that differ by an amount of 32 or more can be ranked as “better” and “worse”, where the confidence in the ranking increases with increasing numeric difference.

Similarly, differing RSSI values reported by the same device at different times or for different remote correspondents may be compared reliably, as can RSSI values among devices that are known (by vendor and other device-specific model identification) to provide calibrated RSSI estimates. In other cases such comparisons are at best approximate, somewhat similar to RSQI though presumably more closely approximating the actual strength of received signaling at the reporting device. In particular, magnitude ordering among reports from the same reporting device are always reliable in terms of their ordering and approximate magnitude of difference.

9.1.15.3 Accumulating and reporting diagnostic information

The system manager establishes a DL communication relationship between a DLE and its neighbor by adding an entry to the DLE’s dlmo.Neighbor attribute. Each such entry specifies a level of diagnostics to be collected, through the field dlmo.Neighbor[ ].DiagLevel. For each neighbor, diagnostics may be collected at a baseline level, or at a detailed level including clock diagnostics.

Per-channel diagnostics are accumulated and consolidated for all neighbors, in the attribute dlmo.ChannelDiag.

When the dlmo.Neighbor[ ].DiagLevel field is set for a particular neighbor, the DLE shall create corresponding entries in the read-only attribute dlmo.NeighborDiag. NeighborDiag values are accumulated from the time that the dlmo.NeighborDiag entry is created.

Three mechanisms are provided for reporting diagnostic information contained in dlmo.NeighborDiag and dlmo.ChannelDiag:

- The health reports concentrator object (HRCO), described in 6.2.7.7, can be configured to report any attribute in the DLE on a periodic basis. dlmo.NeighborDiag entries and dlmo.ChannelDiag can be reported through that mechanism.
- Diagnostic information can be retrieved at any time by the system manager, by reading the applicable attributes.
- Diagnostic information can be reported by the DLE on an exception basis, through the DL_Connectivity alert.

Diagnostics include a combination of levels, such as RSSI, and counters, such as a count of acknowledgments.

Levels are accumulated as exponential moving averages (EMAs). The level is initialized with the first data value, after which each new data value is accumulated into the EMA level as follows, where:

$$\text{EmaLevel}_{NEW} = \text{EmaLevel}_{OLD} + (\alpha / 100) \times (\text{NewData} - \text{EmaLevel}_{OLD})$$

The smoothing factor α is expressed as an integer in the range of 0..100, representing a percentage; it is configured by the system manager through the attribute dlmo.SmoothFactors (see 9.4.2.25).

Counters in dlmo.NeighborDiag are accumulated as ExtDLUint unsigned integers, which internally are 15-bit integers. When a counter reaches its maximum value, of 32 767 (0x7FFF), it shall “stick” and continue to report that maximum value. Counters shall be reset to zero whenever the row is reported through the HRCO or retrieved through a read operation. Reporting the value through the DL_Connectivity alert shall not reset any counters.

9.1.15.4 Radio silence

The DLE can be configured to transmit only when actively participating in a D-subnet. This behavior is configured by the dlmo.RadioSilence attribute, which designates a timeout period for D-subnet participation, in seconds. For example, if the dlmo.RadioSilence attribute is set to the default of 600 s (10 min), the DLE silences its radio transmitter 10 min after losing communication with the D-subnet. When all DLEs on a D-subnet are configured for radio silence, it is possible to disable the D-subnet entirely, even if some DLEs do not receive an explicit command to disable communications.

When a valid time update is accepted by the DLE from an advertisement or an acknowledgment, the DLE internally records the current time as the radio silence time reference. If the DLE does not accept another time update in the subsequent time period designated by dlmo.RadioSilence, the DLE shall become silent by ignoring all of its configured transmit links, including solicitations. In the radio silent state, the DLE continues to operate its radio receiver as per its scheduled receive links, but without transmitting acknowledgments in the absence of a clock update.

For example, suppose the DLE receives a time update at 01h:02m:03s, and dlmo.RadioSilence is set to 600 s (10 min). If the DLE does not receive another time update by 01h:12m:03s, i.e., 10 min later, it will silence its radio at that time.

If dlmo.RadioSilence is configured as zero, the feature is disabled.

Radio silence is the default. The default D-subnet discovery procedure does not use solicitations, and dlmo.RadioSilence defaults to 600 s.

Support of the dlmo.RadioSilence attribute is required for all DLEs.

The radio silence profile limits the permitted range of the dlmo.RadioSilence attribute. The radio silence profile is reported to the system manager on joining through dlmo.DeviceCapability. A DLE with the radio silence profile shall reject updates to dlmo.RadioSilence that are greater than 600 s, thus ensuring that such a DLE will never spontaneously transmit a DPDU once it has lost contact with the D-subnet for 600 s.

Temporary radio silence can be accomplished with another attribute, dlmo.RadioSleep. When dlmo.RadioSleep is set to a positive value, the DLE treats all links, including receive links, as idle for the designated number of seconds. Activation of dlmo.RadioSleep shall be slightly delayed to allow for transmitting an AL acknowledgment for the DMAP APDU that causes the attribute to be set. When the sleep period is over, dlmo.RadioSleep is automatically reset to zero, indicating that the feature is disabled.

9.1.15.5 Radio transmit power

This standard provides the system manager with a degree of control over radio transmit power, through the attribute dlmo.RadioTransmitPower.

dlmo.RadioTransmitPower is used to control the DLE's radio transmit power level, in dBm EIRP. It defaults to the DLE's maximum supported power level, and is always constrained by dlmo.CountryCode (9.1.15.6) to the regulatory constraints of the locale of use. This constrained default value is also reported to the system manager during the joining process through dlmo.DeviceCapability.

When dlmo.RadioTransmitPower is changed by the system manager, the DLE shall not transmit at an output power level in excess of dlmo.RadioTransmitPower.

In addition, a DLE may autonomously calibrate its output power level to the minimum level needed to maintain reliable connectivity. To enable this, the DLE supports the echoing of signal quality information in acknowledgments, so that an implementation can calibrate the received signal quality at various power levels. See 9.3.5.5.

NOTE An accurate calculation of the DLE's actual output level from correspondent reports of received RSSI depends on design information for the reporting correspondent DLE that is not known to the self-calibrating DLE. Those factors include the correspondent DLE's receiver noise floor and minimum receive sensitivity. Thus any self-adjustment of transmit levels based on received RSSI is at best approximate, particularly when the corresponding DLEs' RF subsystems do not share a common design (such as when they are from different manufacturers).

It is possible for DLEs to provide local correction of the RSSI values that they report to account for the influence of their own receiver's noise floor and minimum receive sensitivity. Such self-correction, which is not addressed by the IEEE 802.15.4 standard, typically can be performed in a piecewise-linear manner. The resulting reported values are considered to meet the RSSI requirements of both this standard and those of IEEE 802.15.4, even though they are slightly adjusted from the measurements of the RF subsections of implementing DLEs (since those actual measurements do not take those other relevant characteristics of the RF subsystem into account). See also 9.1.15.2.

9.1.15.6 Country code

The provisioning DLE and/or the system manager can inform the DLE being provisioned of regulatory considerations through its dlmo.CountryCode attribute, Table 103, which is a 16-bit packed structure consisting of a 10-bit country code and six Booleans:

- Bits 0..9 provide a 10-bit country code as an Unsigned10 integer, using ISO 3166-1 numeric three-digit country codes.

- Bits 10..15 specify a six-element Boolean array:
  - Bit10 (Index 0), FCC, indicates whether US FCC rules apply. A DLE shall operate in compliance with US FCC rules when Index0 (Bit10) is TRUE.
  - Bit11 (Index 1), ETSI, indicates whether EU rules (ETSI standards) apply. A DLE shall operate in compliance with EU rules when Index1 (Bit11) is TRUE.
  - Bit12 (Index 2), LP, indicates whether a 10 dBm EIRP limit applies. A DLE shall limit its emissions to ≤ 10 dBm EIRP when Index2 (Bit12) is TRUE.
  - Bit13 (Index 3), LBT, indicates whether the DLE shall operate under adaptivity rules, using LBT to sense the channel when initiating a transaction, ceasing use of the slot if activity is detected: FALSE=non-adaptive, TRUE=adaptive.
  - Bit14 (Index 4), FHSS, indicates whether the DLE shall operate under frequency-hopping spread-spectrum rules: FALSE=not-FHSS-rules, TRUE=FHSS-rules.
  - Bit15 (Index 5), Locked, indicates whether the value of this attribute is fixed while the DLE is operational. Once this “sticky” bit is set, any subsequent attempt to modify the dlmo.CountryCode attribute shall be rejected except when the DLE is reset to the factory default state during (re)provisioning.

Table 103 – CountryCode

Octet number	Bits							
	7	6	5	4	3	2	1	0
1	Mode						ISO 3166-1 CountryCode bits 9..8	
	Locked	FHSS	LBT	LP	ETSI	FCC		
2	ISO 3166-1 CountryCode bits 7..0							

When Bit11 and Bit13 (ETSI and LBT) are both TRUE, each D-transaction shall begin with an LBT observation interval of at least 20 us, using CCA Mode 1, thus supporting modes as described in Annex V, 3) and Annex V, 6).

NOTE 1 CCA Mode 3 is also acceptable under ETSI EN 300 328 v1.8.1 when it is implemented as the AND of CCA Mode 1 and CCA Mode 2, but not when it is implemented as the OR of CCA Mode 1 and CCA Mode 2. See 9.1.9.4.3.

When Bit11, Bit13 and Bit14 (ETSI, LBT and FHSS) are all TRUE, operation switches momentarily to the non-adaptive rules of ETSI ETSI EN 300 328 v1.8.1 while sending an ACK/NAK DPDU (as short control signaling) within a transaction and for the immediately following Tx-gap-time of EN-mandated non-transmission, thus supporting the mode described in Annex V, 6).

Bit15 (Locked) supports device operation (when TRUE) in regulatory regimes that prohibit the ability to reconfigure a device in such a way that it would violate regulatory restraints, while still supporting devices (when FALSE) on mobile platforms such as ships and trains that may cross regulatory jurisdictional boundaries, and while still permitting (via the reprovisioning exception) the repair or refurbishment of devices with subsequent resale into or reuse in markets where other regulations apply.

When no specific country of intended use has been identified, the default for dlmo.CountryCode shall be 0x3C00, indicating that a device in the default state should comply with US FCC rules, EU rules, the < 10 dBm EIRP limit, and be classified as an adaptive non-FHSS device. See 5.2.5 and Annex V.

NOTE 2 This default value ensures that the equipment, before it has been provisioned, meets the regulatory requirements of most regions in which it might be deployed, and in particular as such rules would apply to the three-channel default configuration used for out-of-the-box over-the-air provisioning. Such constraint enables the device to participate in short-range provisioning over the Type A wireless medium, at which point the dlmo.CountryCode attribute would be changed to reflect the intended regulatory regime that applies to the device’s initial (and usually only) locale of deployment.

9.1.16 DLE roles and options

The DL specified by this standard is designed with the general goal of constraining the range of construction options for a conforming device, while enabling flexible and innovative system solutions.

The DL framework does not require that all DLEs be equivalent. For example, some routers, designed as dedicated infrastructure devices, might have a continuous source of energy, powerful processors, and essentially unlimited memory capacity. In contrast, some field instruments may have low-capacity batteries and may lack routing capability.

These distinctions among DLEs are covered in three general ways in this standard:

- memory capacity;
- DLE capabilities; and
- DLE roles.

Every DLE has a limited amount of memory that is available for DL operations, and the system manager needs knowledge of these limitations in order to configure the DLE and balance the D-subnet operation. DLE DL memory is not reported as a single block, but rather as specific capacities of memory for specific purposes. For example, each indexed OctetString attribute supports a limited number of entries, with the capacity available to the system manager as metadata. Similarly, buffer capacity for Data DPDU forwarding is reported by the DLE on startup.

Certain DLE capabilities are also reported on startup. For example, the DLE reports the stability of its own clock, as well as a list of radio channels that it can support legally. DLE capabilities reported with the join request are enumerated in 9.4.2.23.

DLE roles describe the general capabilities of a given DLE configuration. For example, a DLE may be capable of routing or not. Distinctions of this type have various implications throughout the DLE, in terms of minimum memory capacity, DLE capabilities, and support of various features. The DLE simply reports which roles it supports, and the system manager is then responsible for mapping this into a portfolio of DLE capabilities. Standard mappings between roles and minimum capabilities are provided in Annex B.

9.1.17 DLE energy considerations

Devices have different levels of available energy. One device may have a continuous energy source. Another device may have a large battery, but may need most of that energy capacity for running a sensor. Yet another device may use energy scavenging as its primary energy source. Different battery chemistries have different characteristics, a given battery chemistry may provide different performance depending on the supplier, and a battery's capacity may vary depending on environmental factors. New battery technologies are likely to emerge with currently unknown performance characteristics. One application might need a 20-year battery life, while a different application might tolerate a 6-month life.

The DLE may be configured by the system manager to consume different amounts of energy. The DLE consumes energy in two general ways:

- The DLE consumes energy by providing wireless service to its own applications. When a DLE establishes a contract to transmit data every 5 s, the DLE consumes a corresponding amount of energy.
- The DLE consumes energy acting as a router on behalf of neighboring DLEs. A DLE may be configured to transmit advertisements every 10 s. A DLE may be configured to operate its receiver almost continuously, listening for solicitations. The D-subnet may be configured so that a DLE forwards up to 100 DSDUs per minute. All of these scenarios consume energy.

The DLE reports a general sense of its capacity to support DL routing operations in certain fields of the dlmo.EnergyDesign attribute. This attribute is reported through the dlmo.DeviceCapability attribute.

dlmo.EnergyDesign indicates the device's designed energy capacity to handle DL operations. This attribute is constant over the life of the device and reflects the device's design, not its current state. A system manager should configure a DLE within these stated energy limitations:

– EnergyLife indicates the device's energy life by design. A positive value provides energy life in days; a negative value provides energy life magnitude in hours. A value of 0x7FFF indicates a continuous power source and no constraining device energy limitations. Other EnergyDesign fields describing DLE energy capacity are based on this target energy life. Configuration of the DLE beyond these stated energy capacities will likely reduce the device's energy life.
– ListenRate indicates the DLE's energy capacity on average, in seconds per hour, to operate its radio's receiver. ListenRate includes time to receive Data DPDUs for the DLE's own application contracts, plus Data DPDUs being forwarded by the DLE on behalf of other DLEs.
– TransmitRate indicates the DLE's energy capacity, in Data DPDUs per minute, to transmit Data DPDUs on its own behalf and to forward Data DPDUs on behalf of its neighbors.
– AdvRate indicates the DLE's energy capacity, in Data DPDUs per minute, to transmit dedicated advertisement (or solicitation) Data DPDUs.

EnergyDesign is a constant, and does not reflect the changing state of a device's energy source. The dlmo.EnergyLeft attribute is a dynamic read-only attribute that can be used to report the device's remaining energy capacity. A positive value indicates the remaining life in days, and a negative value indicates the magnitude of the remaining life in hours. A value of 0x7FFF indicates that the feature is not supported. dlmo.EnergyLeft is reported on startup through dlmo.DeviceCapability, and may also be reported periodically through the HRCO.

9.2 DDSAP

9.2.1 General

The DDSAP supports the multi-hop conveyance of a DSDU (e.g., and NPDU) between DLEs in a D-subnet.

DD-DATA.request takes a DSDU from the NLE, prepends a Data DPDU header, and adds it to the message queue. DD-DATA.confirm subsequently reports whether the DSDU was successfully conveyed to a neighboring DLE in the D-subnet.

DD-DATA.indication indicates the receipt of a Data DPDU that has reached its final destination within the D-subnet, and passes its DSDU to the NLE.

All interfaces between the DLE and adjacent layer entities or management entites are internal interfaces within the device, and thus are unobservable. Therefore they are strictly notional and not subject to standardization.

9.2.2 DD-DATA.request

DD-DATA.request is a primitive that accepts DSDU from the NL, selects the route through the D-subnet, and places a corresponding Data DPDU on the DLE's message queue.

The semantics of the DD-DATA.request primitive are as follows:

DD-DATA.request (
SrcAddr,
DestAddr,
Priority,
DE,
ECN,
LH,
ContractID,
DSDUSize,
DSDU,
DSDUHandle)

Table 104 describes the parameters for DD-DATA.request.

Table 104 – DD-DATA.request parameters

Parameter name	Parameter type
SrcAddr (DL source address)	Type: DL16Address or EUI64Address
DestAddr (DL destination address)	Type: DL16Address or EUI64Address
Priority (priority of the payload)	Type: Unsigned4
DE (discard eligible)	Type: Unsigned1
ECN (explicit congestion notification)	Type: Unsigned2
LH (last hop, NL)	Type: Unsigned1
ContractID (ContractID of the payload)	Type: Unsigned16 or null
DSDUSize (payload size)	Type: Unsigned8
DSDU (number of octets as per DSDUSize)	Type: Octets
DSDUHandle (uniquely identifies each invocation of this primitive)	Type: Abstract

DD-DATA.request parameters include:

- SrcAddr is the source address of the NSDU. It is normally the DL16Address alias of the NSDU's source IPv6Address, except when it is the EUI64Address of an unjoined DLE. Subnet ID is implicit, based on dlmo.SubnetID.
- DestAddr is the destination address of the NSDU. It is normally the DL16Address alias of the NSDU's destination IPv6Address, except when it is the EUI64Address of an unjoined DLE. Subnet ID is implicit, based on dlmo.SubnetID.
- Priority is copied to the DROUT subheader and indicates the Data DPDU's priority in DLE message queues.
- DE is copied to the DADDR subheader. DE=1 indicates that the DSDU is eligible to be discarded from a message queue in favor of an incoming Data DPDU with DE=0, and of equal or higher priority.

NOTE The expression "is eligible" does not mean "is mandatory". Such discard is an implementation option.

- ECN is copied to the DADDR subheader. See 9.1.9.4.5 for a discussion of ECN.
- LH is copied to the DADDR subheader. A value of 1 indicates that the DSDU entered the D-subnet through a backbone router, and therefore shall not exit the DL through a backbone router to avoid circular routes at the NL. This enables the NL to elide the IPv6 hop limit field. Logically, LH is carried by the DL on behalf of the NL, and LH shall not be changed by the DL.
- ContractID may be used by the DLE in route selection, as discussed in 9.1.6.5.
- DSDUSize indicates the number of octets contained in the Data DPDU payload.
- DSDU is the set of octets forming the payload. It may be implemented as a pointer to memory that is shared among layers.
- DSDUHandle is an abstraction that connects each invocation of DD-DATA.request with the subsequent callback by DD-DATA.confirm.

9.2.3 DD-DATA.confirm

DD-DATA.confirm is a primitive that reports the results of a request to transmit a DSDU that was previously placed on the DLE message queue by DD-DATA.request.

Table 105 describes the parameters for DD-DATA.confirm.

Table 105 – DD-DATA.confirm parameters

Parameter name	Parameter type
DSDUHandle (identifier for the payload)	Type: Abstract
Status (see Table 106)	Type: Unsigned

Table 106 specifies the value set for the status parameter.

Table 106 – Value set for status parameter

Value	Description
SUCCESS	Operation was successful
FAILURE	Operation was unsuccessful; operation timed out

NOTE Error handling between the DLE and colocated NLE is an internal device matter, not visible across any observable interfaces, and therefore is not standardized.

9.2.4 DD-DATA.indication

DD-DATA.indication is a virtual primitive that indicates the receipt of a DSDU. A Data DPDU does not trigger a data indication until it reaches its destination on the D-subnet.

The semantics of the DD-DATA.indication primitive are as follows:

DD-DATA.indication(
SrcAddr,
DestAddr,
Priority,
DE,
ECN,
LH,
DSDUSize,
DSDU)

Table 107 describes the parameters for DD-DATA.indication.

Table 107 – DD-DATA.indication parameters

Parameter name	Parameter type
SrcAddr	Type: DL16Address or EUI64Address
DestAddr	Type: DL16Address or EUI64Address
Priority (priority of the payload)	Type: Unsigned4
DE (discard eligible)	Type: Unsigned1
ECN (explicit congestion notification)	Type: Unsigned2
LH (last hop, NL)	Type: Unsigned1
DSDUSize (payload size)	Type: Unsigned8
DSDU (number of octets as per DSDUSize)	Type: Octets

DD-DATA.indication parameters include:

- SrcAddr is the source address of the NSDU. It is normally the DL16Address alias of the NSDU's source IPv6Address, except when it is the EUI64Address of an unjoined DLE. D-subnet ID is implicit, based on dlmo.SubnetID.
- DestAddr is the destination address of the NSDU. It is normally the DL16Address alias of the NSDU's destination IPv6Address, except when it is the EUI64Address of an unjoined DLE. D-subnet ID is implicit, based on dlmo.SubnetID.
- Priority is included in the DROUT subheader and may be used by the NL for subsequent routing. ContractID, if required by the NL, is not carried within the Data DPDU header.
- DE provides the value of the DE bit copied from the incoming DADDR subheader.
- ECN provides the value of the ECN bit copied from the incoming DADDR subheader, and corresponds to the ECN bit described in IETF RFC 3168. See 9.1.9.4.5 for a discussion of ECN.
- LH provides the value of the LH bit copied from the incoming DADDR subheader.
- DSDUSize indicates the number of octets contained in the Data DPDU payload.
- DSDU is the set of octets forming the payload. It may be implemented as a pointer to memory that is shared among layers.

9.3 Data DPDUs and ACK/NAK DPDUs

9.3.1 General

The structure of DPDUs used by this standard is shown in Figure 88.

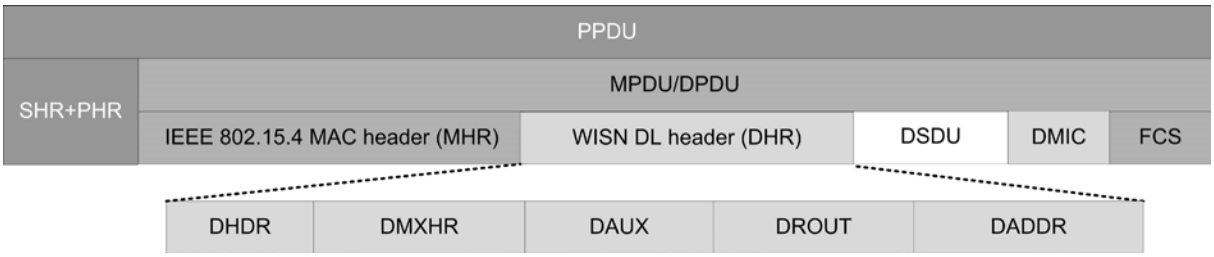


Figure 88 – PhPDU and DPDU structure

The DPDU reflects the multi-layer PDU structure described in 9.1.4, including:

- MAC header (MHR): The MHR is a data structure modeled on that of IEEE 802.15.4, as specified in 9.1.4 and 9.1.5, which includes frame-format information and D-subnet addressing information. The FCS, at the end of the DPDU, is logically associated with the MHR.
- DPDU header (DHR): The DL header information follows the MHR. Subheaders within the DHR include:
  - DHR header (DHDR): DHDR includes settings for various DLE selections and a version number.
  - DHR MAC extension subheader (DMXHR): Additional fields, not specified by IEEE 802.15.4, that are needed to send a Data DPDU to an immediate neighbor and to receive an immediate ACK/NAK DPDU. The DMXHR includes information about the cryptographic integrity and confidentiality measures that apply to the DPDU. The DMIC following the DSDU is logically associated with the DMXHR.
  - DHR auxiliary subheader (DAUX): Some DPDUs include auxiliary information to facilitate neighbor discovery, time propagation, information exchange, and command exchange among immediate neighbors. The DAUX subheader is frequently absent. A non-null DAUX field shall be included in dedicated

advertisement or solicitation DPDUs, or alternatively it may be embedded in unrelated Data DPDUs.

– DHR routing subheader (DROUT): The DROUT field contains information needed to route the contained DPDU payload through the D-subnet. A non-null DROUT field shall include a Data-DPDU priority class and forwarding limit, plus either GraphID or source routing information.

– DHR address subheader (DADDR): The DADDR field contains the source and destination endpoint D-addresses within the D-subnet, along with the NL fields ECN, DE, and LH, all of which are conveyed by and visible to the DL.

- DSDU: The DPDU's higher-layer payload is a single 6LoWPAN NPDU as defined in Clause 10, which is passed to the DLE as a DSDU. The payload is conveyed transparently within the D-subnet.
- DMIC: The DMIC, found near the end of the DPDU, is logically associated with the DMXHR. The DMIC is a cryptographically-strong integrity code that permits determination that the received DPDU

  – was originated by a device that shares the relevant encryption key, and

  – was unaltered before reception.

NOTE In some cases the relevant DMIC encryption key is static and published, enabling forgery by an uninformed attacker.

- FCS: The FCS, found at the end of the DPDU, is logically associated with the MHR. The FCS is a trivially-forgeable integrity code that enables detection of PhL-induced DPDU errors.

Some classes of DPDUs that are generated by the DLE, such as dedicated advertisements and solicitations, have null DROUT, DADDR and DSDU fields.

9.3.2 Octet and bit ordering

9.3.2.1 General

Except in the DL, this standard uses most significant octet first (MSB or big-endian) transmission and documentation conventions, following the precedent set by ISO, IEC, IETF, and many others. That is:

- for multi-octet values, the most significant octet is transmitted first; and
- octet documentation shows bit 7 on the left and bit 0 on the right.

However, IEEE 802.15.4 uses the least significant octet first (LSB or little-endian) conventions. That is:

- for multi-octet values, the least significant octet is transmitted first.

NOTE The IEEE specification is not entirely consistent on this point: IEEE 802.15.4:2011 security subheaders use MSB transmission and documentation conventions.

Bit transmission order within an octet is handled at the PhL. Within the DL the discussion of MSB and LSB is limited to the ordering of octets.

As a result, the DL is unavoidably mixed-endian, with some sections using big-endian and others using little-endian bit ordering.

Generally, the standard DPDU headers follow IEEE 802.15.4:2011 conventions, as follows:

– This standard, except for DL and MAC headers, follows MSB conventions.

– Standard DL and MAC headers follow LSB conventions, with some clearly indicated exceptions in the DPDU header.

– Within the DPDU, security subheaders follow MSB conventions, following the IEEE 802.15.4 precedent.
– All fields within the DPDU header are documented showing bit 7 on the left and bit 0 on the right, following the convention of this standard.
– DLMO attributes, accessible to the system manager through the DMAP, are AL information, and as such generally use MSB conventions. Some exceptions are made for fields that interact directly with DPDU headers that use the LSB convention.

LSB octet ordering is explicitly noted in various parts of the DL specification. By convention, octet 0 is the least significant octet. LSB indicates that the least significant octet (octet 0) is transmitted first, and the most significant octet (octet n) is transmitted last. When not specified, the reverse ordering is used for transmission.

9.3.2.2 Extensible DL unsigned integers

The DL specification uses a construct called ExtDLUint for compressed transmission of unsigned integers. This is not a type used in other standards, and as such ExtDLUint only appears in DPDU headers and within DL-defined octet strings. It is used to indicate compressed encoding of a 15-bit unsigned integer; it is not used in conveying other data. Since this type is not used outside of the DL, it is not specified as a standard AL-supported data type.

An ExtDLUint shall be transmitted as one octet when its value is in the range of 0..127, and as two octets when its value is in the range of 128..32 767, with encoding as shown in Table 108 and Table 109. Bit 0 in the first octet indicates whether one or two octets are transmitted. Octet ordering is always as shown here, with the size indicated in bit 0 of the first octet transmitted.

Table 108 – ExtDLUint, one-octet variant

Octets	Bits							
	7	6	5	4	3	2	1	0
1	2^6	2^5	2^4	2^3	2^2	2^1	2^0	Selection = 0

Table 109 – ExtDLUint, two-octet variant

Octets	Bits							
	7	6	5	4	3	2	1	0
1	2^6	2^5	2^4	2^3	2^2	2^1	2^0	Selection = 1
2	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7

9.3.3 Media access control headers

9.3.3.1 General

This standard uses a MAC header format whose component data structures are compliant with the detailed structure and field coding of IEEE 802.15.4:2011, 5.2.1 and 5.2.2.2, followed by extensions that are particular to this standard. Only the Data DPDUs of IEEE 802.15.4:2011, 5.2.2.2, are used by this standard.

This standard does not use IEEE 802.15.4 security. Instead, similar security is handled in the DMXHR. DPDU security in this standard is similar to IEEE 802.15.4 security; the main difference is that this standard incorporates the DLE's shared sense of time and other usage-context-specific information in the cryptographic nonces, resulting in more compact DPDUs

and improved resistance to replay and misdirection attacks. To facilitate that improved resistance to attack, the DPDU sequence numbers of this standard are derived from different content sources than that specified by IEEE 802.15.4:2011, 5.2.1.2.

The ACK/NAK DPDUs of IEEE 802.15.4:2011, 5.2.3, cannot be reliably distinguished from a similarly-timed identical DPDU sent by a device that is not the intended recipient. Such indistinguishability can arise due to concurrent activity on the same physical channel by other devices than the intended recipient, for example in other nearby networks, or due to an attacker's deliberate spoofing of the ACK/NAK DPDU after jamming reception of the Data DPDU of the transaction. Therefore this standard uses short, authenticatable Data DPDUs to implement similar but extended ACK/NAK functionality. When the destination and source MAC addresses of such ACK/NAK DPDUs are those of the source and destination MAC addresses, respectively, of the soliciting Data DPDU, there is no need to convey those addresses explicitly. Thus this standard permits such ACK/NAK DPDUs to suppress both MAC address fields, which contradicts the constraint of IEEE 802.15.4:2011, 5.2.1.1.8.

NOTE Although the amendments of IEEE 802.15.4 were intended to cover similar issues to those that led to the just-described variances, they often do so in ways that are incompatible with the ISA 100.11a:2011 standard on which this international standard is based, and thus are also incompatible with this standard, which strives to maintain compatibility with deployed equipment that uses that ANSI/ISA standard.

9.3.3.2 Media access control header

The format of the subset of the standard MHR specified in IEEE 802.15.4:2011, 5.2.1, and IEEE 802.15.4:2011, Figure 35, as used by this standard, is summarized in Table 110.

Table 110 – Data DPDU MHR

Number of octets	bits							
	7	6	5	4	3	2	1	0
2	Frame control (LSB ordering)							
1	Sequence number							
0 or 2	PAN ID							
0, 2, or 8	Destination address							
0, 2, or 8	Source address							
NOTE The PAN ID, Destination address and Source address fields are each transmitted in LSB order.								

The size of the MHR is usually 9 octets, including a PAN ID and two DL16Addresses, with these exceptions:

- Solicitations data DPDUs have a null (zero-length) PAN ID and two null MAC addresses, so the MHR is 3 octets.
- Advertisement data DPDUs have a null destination MAC address, so the MHR is 7 octets.
- Other data DPDUs to or from an unjoined DLE have one DL16Address and one EUI64Address, so an MHR addressed to or from an unjoined DLE is 15 octets.

NOTE 1 The default DPDU payload capacity, dlmo.MaxDsduSize, is based on an MHR size of 15 octets, providing a basis for making fragmentation decisions for unjoined DLEs.

- ACK/NAK DPDUs, which are used for immediate acknowledgments (short control signaling) have a null PAN ID, a null destination MAC address, and a source MAC address that is either
  - null (zero-length), so the MHR is 3 octets; or
  - a DL16Address, so the MHR is 5 octets; or
  - an EUI64Address, so the MHR is 11 octets.

NOTE 2 The PAN ID is suppressed because the ACK/NAK DPDU's security authentication serves to reject any ACK/NAK DPDU intended for a different device, whether on the same PAN or a different PAN.

As shown in Table 110, fields include:

a) Frame control. For this field, subfields are as specified in IEEE 802.15.4:2011, 5.2.1.1:
   - FrameType shall be Data.
   - SecurityEnabled shall be FALSE, because IEEE 802.15.4 is not used.

     NOTE 3 The extended security of this standard is handled in the DMXHR.
   - FramePending shall be FALSE.
   - AckRequest shall be FALSE.

     NOTE 4 The readily-spoofed IEEE 802.15.4 immediate-acknowledgement DPDU type is not used by this standard.
   - When both a destination D-address and a source D-address are included, PAN ID compression shall indicate that the same PAN ID is used for both D-addresses. Otherwise, PAN ID compression shall be FALSE (because such compression only applies when there are two D-addresses).
   - MAC addresses are usually DL16Addresses, with exceptions as described below. EUI64Addresses are used by a DLE when joining a D-subnet. Destination addresses are omitted in dedicated advertisements and in solicitations.
   - The frame version shall be 0x01.

b) Sequence number. Used by the DSC, as described in 7.3.2.4.10.

   NOTE 5 IEEE 802.15.4 requires that each DLE increment its sequence number after each use, so that the sequence number is unique for all Data DPDUs and ACK/NAK DPDUs originated by that DLE. However, this standard uses the "sequence number" field for a somewhat different purpose and so provides an alternate method (other than cyclic sequentiality) of ensuring that cryptographic nonces generated by each real device are unique within the operational lifespan of the cryptographic key with which they are employed.

   NOTE 6 In this standard both the DLE and the TLE generate nonces. The provisions referred to in NOTE 5 ensure that the two sets of generated nonces are disjoint.

c) PAN ID shall match dlmo.SubnetID and shall be absent in solicitation Data DPDUs. If the DPDU conveys both a source and a destination MAC address, the PAN ID shall be the destination's PAN ID (with the source PAN ID inferred to be identical). If there is no destination address, such as in an advertisement Data DPDU, the PAN ID shall be a source PAN ID. In a solicitation Data DPDU, where there is neither a source nor a destination address, this field shall be null (elided). This field shall be null (elided) in all ACK/NAK DPDUs. See 9.1.10.2.

d) Destination address is normally a DL16Address alias for an IPv6Address. An EUI64Address shall be used to address DLEs that have not yet received a DL16Address. The destination address shall be absent in dedicated advertisement Data DPDUs, in solicitation Data DPDUs, and in ACK/NAK DPDUs.

e) Source address is normally a DL16Address alias for an IPv6Address. An EUI64Address shall be used to identify DLEs that have not yet received a DL16Address. The source D-address shall be absent in solicitation Data DPDUs, and in ACK/NAK DPDUs where the D-address would be identical to the destination D-address of the received Data DPDU that initiated the transaction.

9.3.3.3 Data DPDU subheader

The structure of the DHDR for a Data DPDU is shown in Table 111.

Table 111 – Data DPDU DHDR

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	ACK/NAK DPDU expected	Request signal quality in ACK DPDU	Request EUI64Address in ACK DPDU	Include DAUX	DMXHR includes slow-channel-hopping-offset	Clock recipient	DL version Always 01	

This DHDR is always 1 octet.

As shown in Table 111:

- Bit 7 indicates whether ACK/NAK DPDUs are expected from the explicitly or implicitly addressed recipients.
- Bit 6 indicates whether the receiving DLE should report signal quality information in the ACK/NAK DPDU.
- Bit 5 indicates whether the receiving DLE should include its EUI64Address in the ACK/NAK DPDU. This setting shall be used by the sender whenever an acknowledgment is requested (bit 7 value of 1), and no EUI64Address for the neighbor exists in dlmo.Neighbor. See 9.1.10.1.

NOTE Bit 7 is meaningful only for the initial Data DPDU of a transaction. Bits 6 and 5 are meaningful only when Bit 7 is meaningful and has the value TRUE.

- Bit 4 indicates the presence or absence of a DAUX subheader in the Data DPDU.
- Bit 3 indicates whether a slow-channel-hopping-offset is included in the DMXHR. This value shall be included in unicast Data DPDUs where slow-channel-hopping is used. See 9.1.9.2.4.
- Bit 2 indicates whether the transmitting DLE is a DL clock recipient. This is an implicit request to the receiver to include a clock correction in the acknowledgment.
- Bits 0..1 indicates the DL version number. A value of 0x01 shall be used, with 0x10 being reserved for future use. A value of 0x11 is used in the same location in an ACK/NAK DPDU and helps to distinguish a Data DPDU from an ACK/NAK DPDU (see 9.3.4).

9.3.3.4 DPDU MAC extension subheader

A DMXHR following the DHDR is summarized in Table 112.

Table 112 – Data DPDU DMXHR

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Security control							
1	Crypto Key Identifier							
0..2	Slow-channel-hopping-offset (ExtDLUint)							

NOTE For future PHYs with more than 16 channels, it is likely that the channel number will be added as a virtual field. This is a subject of future standardization, but has been considered in the DMXHR design.

The size of the DMXHR is 2..4 octets. A DMXHR size of 3 octets, corresponding to a slow-channel-hopping rate of about 1,25 s or less, was used to calculate the default dlmo.MaxDsduSize.

As shown in Table 112, attributes include:

- Security control and Crypto Key Identifier. The security fields are left unspecified by the DL, and are set by the DSC. See 9.1.12 for an overview of the relationship between the DL and DSCs. While IEEE allows a Crypto Key Identifier as large as 9 octets, in this standard its size is always 1 octet.
- The slow-channel-hopping-offset specifies the timeslot offset into the slow-channel-hopping period, if necessary to unambiguously identify a timeslot (because the transaction initiator and responder have different local perceptions of the proper timeslot). The presence or absence of this field is indicated in the DHDR. The slow-channel-hopping-offset is described in 9.1.9.4.9.

9.3.3.5 DPDU auxiliary subheader

The DAUX subheader is used for:

- DL neighbor discovery;
- temporarily activating links;
- reporting received signal quality in acknowledgments.

The DAUX subheader, present only when bit 4 of the DHDR octet is set, is described in 9.3.5.

A DAUX size of 0 octets was used to calculate the default dlmo.MaxDsduSize.

NOTE dlmo.MaxDsduSize is used to make fragmentation decisions. The DAUX subheader is usable to activate links in a fragmentation scenario. However, link activation is not possible during the joining process, and as such link activation is never combined with EUI64Addresses. Since the calculation of dlmo.MaxDsduSize includes one EUI64Address, it allows for a DAUX link activation subheader when an EUI64Address is not present.

9.3.3.6 DPDU routing subheader

There are two variants of the DROUT subheader. A compressed variant, 2 octets in size, is used when a single graph is used for addressing. The compressed variant is also used when single-hop routing is used, with the route being implicit in the MAC-level addressing found in the IEEE 802.15.4 MHR. When a series of addresses is needed, an uncompressed variant of the DROUT subheader shall be used.

The DROUT subheader shall be elided in a Data DPDU that has no higher-layer payload, as indicated by a DSDU of zero size.

The compressed variant of the DROUT subheader shall be used in the common case where a single graph, with an index of 255 or less, is used for routing. It is shown in Table 113.

Table 113 – DROUT structure, compressed variant

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Compress=1	Priority				DlForwardLimit		
0..1	DlForwardLimitExt							
1	GraphID							

A DROUT size of 2 octets was used to calculate the default dlmo.MaxDsduSize. MaxDsduSize normally needs to be reduced when source routing is used.

As shown in Table 113, the compressed variant of the DROUT comprises:

- Compress. If this value is set to 1, the compressed variant of the DROUT format shall be used.
- Priority. This shall be set to the Data DPDU's 4-bit priority.
- DlForwardLimit and DlForwardLimitExt (forwarding limit) limit the number of times that a Data DPDU may be forwarded within a D-subnet. If the forwarding limit is less than 7, the value shall be transmitted in DlForwardLimit and DlForwardLimitExt shall be elided. If the forwarding limit is greater than or equal to 7, DlForwardLimit shall be transmitted as 7, and the forwarding limit shall be transmitted in DlForwardLimitExt.

The forwarding limit is initialized by the DL when the route is selected, based on the value of dlmo.Route[ ].ForwardLimit. When a unicast Data DPDU is successfully received by the DL and needs to be forwarded, the Data DPDU shall be discarded if its forwarding limit is zero. If its forwarding limit is positive, the forwarding limit shall be decremented (possibly to zero) and the Data DPDU shall be placed on the message queue.

- GraphID (8 bits). GraphIDs compliant with this standard are 12-bit unsigned integers. In the common case where the route is a single graph ID in the range of 1..255, the compressed variant of the DROUT subheader shall be used. Additionally, the compressed variant is used in single-hop source routing, wherein GraphID=0 shall indicate that the destination is one hop away. Since the single hop destination address can be found in the MHR, it does not need to be repeated in DROUT. GraphID=0 shall be used during the joining process for addressing to and from a neighboring proxy, and is the only way in this standard to indicate a destination EUI64Address in DROUT.

NOTE It is possible for a system manager to configure a circular D-route, with the Data DPDU being forwarded until the ForwardLimit decrements to zero. The LH field in the DL header, described in 9.3.3.7, is not intended to prevent circular routes within a D-subnet.

The uncompressed variant of the DROUT subheader is shown in Table 114.

Table 114 – DROUT structure, uncompressed variant

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Compress=0	Priority				DlForwardLimit		
0..1	DlForwardLimitExt							
1	N (number of entries in routing table)							
2*N	Series of N GraphIDs/addresses (Unsigned16, LSB)							
NOTE Each GraphID/address is transmitted in LSB order.								

As shown in Table 114, the uncompressed variant of the DROUT subheader comprises:

- Compress: If this value is set to 0, the uncompressed variant of the DROUT format shall be used.
- Priority, DlForwardLimit, and DlForwardLimitExt are as described above with Table 113.
- N: This field shall be set to the number of entries in Route. The entries may be a combination of GraphIDs and DL16Addresses. The value of N shall not exceed 15.
- Route: This field shall be set to a series of GraphIDs and/or DL16Addresses, specifying the route, in order, along which the Data DPDU will travel. IETF RFC 4944 limits unicast address ranges to $1..2^{15}$-1. 12-bit GraphIDs in this field shall be represented disjointly from that address range as 0x 1010 gggg gggg gggg, which is the range $10\times2^{12}..11\times2^{12}$-1.

When source routing is used, the DROUT subheader shall be shortened by the DL of intermediate routers as the Data DPDU proceeds along the route, as described in 9.1.6.

The first entry in the DROUT subheader is used to determine the next hop. For example, the route may be specified at the source as <000 123, 000 456, 000 789>. The first hop address, <000 123>, is used to send the Data DPDU to an immediate neighbor. The DROUT subheader, as received by DLE <000 123>, contains the source route <000 123, 000 456, 000 789>. When received, this route is shortened to <000 456, 000 789> (see 9.1.6.3), indicating that address <000 456> is the next hop.

When a graph is specified as the first entry in a source route, the Data DPDU shall follow that graph until it is terminated, as described in 9.1.6.

9.3.3.7 Addressing subheader

The addressing subheader (DADDR) includes NL source and destination addresses, along with three NL fields that are visible to the DL.

The DADDR subheader shall be elided in a Data DPDU that has no higher order payload, i.e., a DSDU of zero size.

The structure of the DADDR subheader is shown in Table 115.

Table 115 – DADDR structure

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	DE	LH	ECN		Reserved=0			
1..2	SrcAddr (ExtDLUint)							
1..2	DestAddr (ExtDLUint)							

A DADDR size of 4 octets was used to calculate the default dlmo.MaxDsduSize, reflecting an assumption that one or the other of the addresses is encoded in one octet.

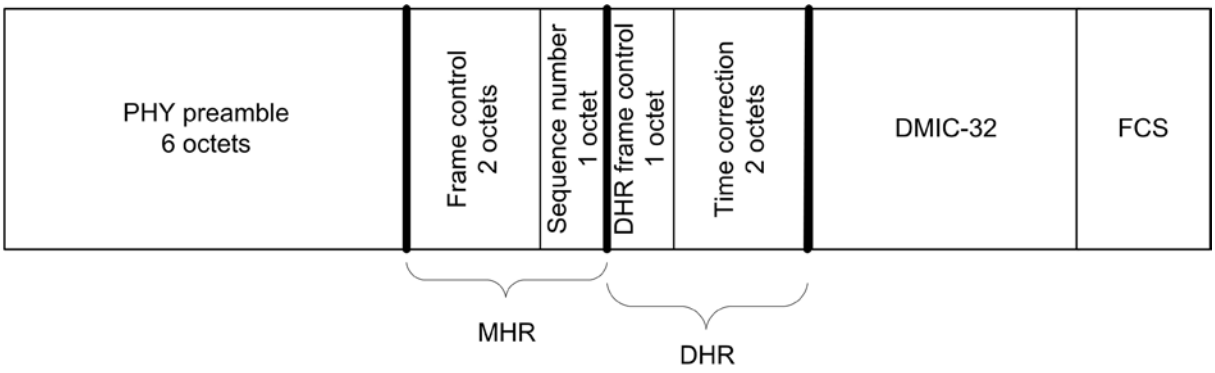
Fields include:

- DiscardEligible (DE) shall be set based on the value provided by the NL through DD-DATA.request. DE=1 indicates that the DSDU is eligible to be discarded from the message queue in favor of an incoming Data DPDU with DE=0, and of equal or higher priority.
- LastHop (LH) shall be set based on the value provided by the NL through DD-DATA.request. This bit is carried by the DL to avoid circular routes at the NL, and does not affect DL behavior.
- ExplicitCongestionNotification (ECN) shall be set based on the value provided by the NL through DD-DATA.request. A router experiencing congestion may set ECN as described in IETF RFC 3168. See 9.1.9.4.5 for a discussion of ECN.
- SrcAddr is set based on the value provided from the NL through DD-DATA.request. If the source D-address is duplicated in the MHR source D-address field, SrcAddr shall be encoded as 0x00. This covers the case, during the joining process, where the source address is an EUI64Address.
- DestAddr is set based on the value provided from the NL through DD-DATA.request. If the destination D-address is duplicated in the MHR destination address field, DestAddr shall be encoded as 0x00. This covers the case, during the joining process, where the destination D-address is an EUI64Address.

By encoding a duplicated DL-address as zero, an octet is compressed on the first and last hop when the address is less than 0x0080, saving energy. Thus a DADDR address encoded as zero references the corresponding DL16Address or EUI64Address in the MHR.

9.3.4 MAC acknowledgment DPDUs

Figure 89 illustrates the structure of ACK/NAK DPDUs.



NOTE This figure includes only the most commonly used fields.

Figure 89 – Typical ACK/NAK DPDU layout

The source address of an ACK/NAK DPDU is the address of the DLE that transmits the DPDU. The destination address is the address of the intended recipient of the DPDU.

Every ACK/NAK DPDU shall be authenticated with a DMIC, but not encrypted. Some fields are virtual, used in creating the DMIC but not actually transmitted.

Although the ACK/NAK DPDU is an IEEE 802.15.4 data frame, it can be distinguished from other such IEEE 802.15.4 data frames based on:

– the ACK/NAK DPDU's timing, following Data DPDU transmission, as specified in 9.4.3.3; and
– bits 1..0 of its DHR frame control field, as specified in Table 118; and
– a virtual field in the ACK/NAK DPDU that echoes the Data DPDU's original DMIC, as specified in Table 117.

As described in 7.3.2.2, the DMIC in an ACK/NAK DPDU uses the same security policy as the original Data DPDU of the D-transaction to which it is a response, with the exception that the ACK/NAK DPDU's DMIC size shall always be 32 bits regardless of the Data DPDU's security policy.

The format of an IEEE 802.15.4 MHR is summarized in Table 116.

Table 116 – ACK/NAK DPDU MHR

Number of octets	Bits							
	7	6	5	6	3	2	1	0
2	Frame control (LSB ordering)							
1	Sequence number							
0	Destination address (null)							
0 or 2	PAN ID							
0, 2, or 8	Source address							
NOTE The PAN ID and Source address fields are each transmitted in LSB order.								

The detailed description of these fields is specified in IEEE 802.15.4. As shown in Table 116, these attributes include:

- Frame control attributes for ACK/NAK DPDUs, as follows:
  - FrameType shall be data.
  - SecurityEnabled shall be disabled, as it is handled in the DHR.
  - FramePending shall be FALSE.
  - AckRequest shall be FALSE.

  NOTE 1 The above bit requests generation of the unsecurable form of immediate acknowledgment offered by IEEE 802.15.4, which is not used by this standard.

  - Source addressing mode shall be 0x00 (i.e., implicit), except for cases described below where the PAN ID and source address are included in the MHR.
  - Destination addressing mode shall be 0x00 (i.e., implicit).
  - FrameVersion shall be 0x01.
- Sequence number, used by the DSC, as described in 7.3.2.4.10. As this standard does not use the unsecurable ACK frame type specified by IEEE 802.15.4, its ACK/NAK DPDU does not carry the sequence number of its preceding Data DPDU. Rather the sequence number shall itself be similar to that of a Data DPDU, as specified in 9.3.3.2, and shall be used in construction of the D-nonce for the ACK/NAK DPDU's DMIC.
- PAN ID, present only when the source address is present (non-null).
- Source address. Normally, a source D-address is not included in an ACK/NAK DPDU, because it matches the destination D-address of the last-received Data DPDU. However, there are two exceptions where it is included:
  - An immediate acknowledger of a received Data DPDU shall include its EUI64Address as the source address of the replying ACK/NAK DPDU's MHR when so requested in the received Data DPDU's DHDR.
  - An immediate acknowledger of a received Data DPDU whose D-address is different than the destination D-address of the last-received Data DPDU shall include its DL16Address as the source address of the replying ACK/NAK DPDU.

  NOTE 2 This second exception occurs in secondary duocast and N-cast acknowledgments.

A prototype DHR following a MHR is summarized in Table 117.

Table 117 – ACK/NAK DPDU DHR

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	ACK/NAK DPDU DHDR							
4 (virtual)	Echoed DMIC of received Data DPDU							
0, 2	Time correction (Unsigned16, LSB) when requested							
0..2	Timeslot offset (ExtDLUint) when needed							
0..3	DAUX subheader usually absent							

As shown in Table 117, attributes include:

- The ACK/NAK DPDU's DHDR is described in Table 118.
- Echoed DMIC of received Data DPDU. For a discussion of handling of this virtual field, see 7.3.2. To unambiguously connect the ACK/NAK DPDU with the Data DPDU to which it is a response, the DMIC of the Data DPDU is included in the ACK/NAK

DPDU's DHR as a virtual field, with octet ordering matching the Data DPDU's DMIC. This virtual field is used to calculate the ACK/NAK DPDU's DMIC, but not transmitted. If the received DMIC is longer than 4 octets, only the initial (leftmost) 4 octets of the DMIC are echoed as a virtual field.

- Time correction (LSB). Used by DL clock sources to correct the time of the DL clock recipient, if it is requested in the received DPDU's DHDR. This 2-octet unsigned value, when included in the ACK/NAK DPDU, echoes the time that the Data DPDU was received. The value, in 2^{-20}s (approximately 0,954 μs), reports an offset from the scheduled start time of the current timeslot in the acknowledger's time base. The reported value is based on the Data DPDU's start time. See 9.1.9.3.2.
- The acknowledger's timeslot offset is provided, when needed, within a slow-channel-hopping period. This value, when included in the ACK/NAK DPDU, indicates the current timeslot in the acknowledger's time base. It shall be included only when the received Data DPDU is received in a different slow-channel-hopping timeslot than is used for the acknowledgment. The first timeslot in a slow-channel-hopping period has an offset of zero. When the corrected timeslot offset is non-zero, the time correction (previous field), when included, shall be an offset of the corrected scheduled timeslot time. Security requires that a DLE's time increases from timeslot to timeslot. Therefore, if the timeslot is corrected to an earlier timeslot by a clock recipient, there shall be an interruption in service, equal to the magnitude of the timeslot correction plus at least one timeslot. See 9.1.9.4.9.
- Auxiliary subheader (DAUX). DAUX may be included in an ACK/NAK DPDU, for the limited purpose of echoing received signal quality (see 9.3.5.5).

In an ACK/NAK DPDU, the DHDR octet communicates the ACK/NAK type and other DPDU substructure information, as shown in Table 118.

Table 118 – ACK/NAK DPDU DHDR

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Clock correction included	Slow-channel-hopping-offset included	ACK/NAK type: 0: ACK 1: ACKwithECN 2: NAK0 3: NAK1		DAUX subheader included	Reserved (=0)	ACK/NAK DPDU (=11)	

The DL protocol version number and MAC security key always match those of the received Data DPDU to which the ACK/NAK DPDU is an immediate acknowledgment, and therefore are not included explicitly in the ACK/NAK DPDU.

Bit content is as follows:

- Bit 7 indicates whether the ACK/NAK DPDU includes clock correction information.
- Bit 6 indicates whether the ACK/NAK DPDU includes a slow-channel-hopping-offset.
- Bits 5..4 indicate the class of the ACK/NAK DPDU:

  0b10: a NAK0 negative acknowledgment, signaling that the Data DPDU was received but could not be acknowledged due to message queue congestion (9.1.9.4.4);

  0b11: a NAK1, signaling that the Data DPDU was received but was not accepted due to a recent history of forwarding problems along the route (9.1.9.4.4);

  0b00: an ACK positive acknowledgment;

  0b01: an ACK positive acknowledgment with an ECN (explicit congestion notification) (9.1.9.4.5).

A router that is signaling ECN in the forward direction should also signal the ECN through ACK/NAK DPDUs when the Data DPDU's priority is 7 or less. A DLE receiving an ECN

through an ACK/NAK DPDU may treat this signal as early notification that it is likely to receive an ECN at upper layers;

- Bit 3 indicates whether the ACK/NAK DPDU includes a DAUX subheader, which may be included in an ACK/NAK DPDU for the limited purpose of reporting received signal quality.
- Bit 2 is reserved and shall be set to zero.
- Bits 1..0 are set to ones (11) to distinguish ACK/NAK DPDUs from other DPDUs.

9.3.5 DL auxiliary subheader

9.3.5.1 General

An auxiliary subheader (DAUX) may be included in any Data or ACK/NAK DPDU. Bits 7..5 of the first octet of the DAUX determine its type, with the subsequent subheader format different for each type. Defined types are:

- Advertisement: type 0 in Data DPDU: provides information needed by new DLEs to synchronize with and join the D-subnet;
- Solicitation: type 1 in Data DPDU: solicits an advertisement from a neighboring DLE;
- Activate link: type 2 in Data DPDU: activates an idle link for a period of time;
- Signal quality: type 3 in ACK/NAK DPDU: reports received signal quality.

All other combinations of type and DPDU-class are reserved for future use.

NOTE Following DL header conventions, DAUX fields use LSB (little-endian) order for transmission. There are some similar structures in the DLMO that use MSB (big-endian) order. For example, superframe structures are specified in both places, using LSB in the DL header and MSB in DLMO attributes.

9.3.5.2 Advertisement auxiliary subheader

9.3.5.2.1 General

Fields within an advertisement DAUX can be grouped logically as:

- advertisement selections;
- time synchronization;
- superframe information;
- join information; and
- integrity check.

Table 119 summarizes the structure of the advertisement DAUX.

Table 119 – Advertisement DAUX structure

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Advertisement selections; see Table 120							
6	Time synchronization; see Table 122							
6..10	Superframe information; see Table 124							
4..10	Join information; see Table 127							
2	Integrity check; see 9.3.5.2.4.4							
NOTE As described in 9.3.5.2.4.2, join information field size is limited to 10 octets.								

The advertising router's D-subnet ID and DL16Address are conveyed through the MAC sublayer, and do not need to be transmitted redundantly within the DAUX.

An advertisement DAUX may be included within a Data DPDU, but shall not be included within an ACK/NAK DPDU.

An advertisement includes information that enables the receiving DLE to create superframes and links to be used during the joining process. This information shall be retained by the DLE at the end of the joining process and, along with DL defaults, constitute a starting database of link scheduling information for the DLE. The same links used for joining are temporarily used for general communications until the system manager provides an alternative configuration.

Attributes set by the DLE based on information in the received advertisement include:

- dlmo.SubnetID is set based on the SubnetID in the advertisement.
- TAI time is synchronized by the advertisement.
- dlmo.Superframe number1 is created with fields copied from the advertisement.
- dlmo.Link number1 is created as a transmit link with fields copied from the advertisement.
- dlmo.Link number2 is created as a receive link with fields copied from the advertisement.
- dlmo.Link number3 may be created as passive scanning receive links with fields copied from the advertisement if provided.
- dlmo.Neighbor is initialized by the DLE with an entry corresponding to the advertising router.
- dlmo.Graph number1 is automatically created by the DLE, to provide access to the advertising router.
- dlmo.Route number1 is automatically created by the DLE as the default route using graph number1.

9.3.5.2.2 Advertisement selections

Table 120 specifies the advertisement selections field in the advertisement DAUX.

Table 120 – Advertisement selections elements

Element name	Element encoding
DauxType	Type: Unsigned3 0=advertisement DAUX
ChMapOv	Type: Unsigned1 0=default
DauxOptSlowHop	Type: Unsigned1 0=default
Reserved (octet alignment)	Type: Unsigned3=0

Table 121 illustrates the structure of the advertisement selections field.

Table 121 – Advertisement selections

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	DauxType=0			DauxChMapOv	DauxOptSlowHop	Reserved=0		

The advertisement selections field is 1 octet. As shown in Table 120, attributes include:

- DauxType. Always set to 0 for an advertisement DAUX. Indicates the DAUX structure in Table 119.
- DauxChMapOv. TRUE indicates that the DauxChMap field is included in the advertisement DPDU. FALSE selects the default channel map of 0x7FFF. This field corresponds to dlmo.Superframe[ ].ChMapOv.
- DauxOptSlowHop. TRUE indicates that D-subnet offers slow channel-hopping, and that DauxChRate is included in the advertisement DPDU. FALSE indicates the default of slotted-channel-hopping.

NOTE Slow-channel-hopping can be used during the joining process as well as thereafter.

- Bits 2..0 are reserved and shall be set to 0.

9.3.5.2.3 Advertisement time synchronization

Table 122 specifies the time synchronization field in the advertisement DAUX.

Table 122 – Advertisement time synchronization elements

Element name	Element encoding
DauxTAIsecond (current TAI time)	Type: Unsigned32 (LSB) Units: 1 s
DauxTAIfraction (fractional TAI second)	Type: Unsigned16 (LSB) Units: 2^{-15} s

Table 123 illustrates the structure of the advertisement time synchronization field.

Table 123 – Advertisement time synchronization structure

Octets	Bits								
	7	6	5	4	3	2	1	0	Interpretation
1	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	DauxTAIsecond; integral part of TAI time with granularity of 1 s
2	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	
3	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
4	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	
5	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}	DauxTAIfraction; fractional part of TAI time with granularity of 2^{-15} s
6	0 reserved	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	
NOTE The above representation is radically different from that of TAINetworkTime (Table 362), with the octet ordering reversed in the first four octets, and the ordering both reversed and shifted one bit in the last two octets.									

NOTE 1 The DauxTAIfraction unit of 2^{-15} s was chosen to match the 32 KiHz very-precise very-low-power “watch” crystals commonly used for the continuous clock hardware of WISN devices.

The time synchronization field is 6 octets. As shown in Table 122, subfields include:

– DauxTAIsecond. Current TAI time in units of 1 s.

– DauxTAIfraction. Fractional TAI second in units of 2^{-15} s, with a range of 0..32 667. Within the TAI second, this indicates the advertisement DPDU's actual start time. (An implementation that actually clocks based on SFD timing should account for a DPDU start time that is nominally 1 octet, or 32 μs, later than the time that the SFD is completely transmitted/received.)

NOTE 2 Although TAI time is normally represented as a 6-octet scaled fixed point binary fraction, modulo 2^{32} s, the above two-part transmittal ordering, where the each part is transmitted separately LSB first and the fractional part has an inserted unused bit at the binary point, does not honor the natural octet ordering of that scaled fixed point fraction.

See 9.1.9 for more information on TAI time and timeslot alignment.

The identity and scheduled timing of the current timeslot can be derived from the Data DPDU's actual start time combined with the join superframe description. See 9.1.9.1.5.

The time in an advertisement shall be an accurate reflection of the advertiser's internal TAI clock to within ±96 μs (i.e., the transmission duration of 3 octets), which is the transmission window jitter presumed for devices conforming to this standard.

An ACK/NAK DPDU to a join request includes a clock correction that may be more precise than the original advertisement, and also more current.

9.3.5.2.4 Advertisement join superframe and links

9.3.5.2.4.1 Advertisement join superframe

NOTE The joining process, including solicitation and advertisements and use of the information conveyed in advertisements, is described in 7.4.

There are three links specified by the advertisement related to neighbor discovery:

- link number1 for sending join requests, addressed to the neighboring advertising router;
- link number2 for receiving subsequent join responses from the advertising router; and
- link number3 for scanning for additional neighbors after the DLE successfully joins the D-subnet.

All of these links refer to superframe number1, which is also specified in the advertisement.

Field names in the advertisement correspond to equivalent fields in dlmo.Superframe and dlmo.Link. Following DL header conventions, LSB octet ordering is used on certain fields that are transmitted using MSB ordering in the superframe itself. To minimize processing requirements and to compress the DAUX subheader, a subset of superframe and link features is supported through the advertisement.

Table 124 specifies the join superframe information field.

Table 124 – Join superframe information subfields

Subfield name	Subfield encoding
DauxTsDur (timeslot duration)	Type: Unsigned16 Units: 2^{-20} s
DauxChIndex (channel-hopping pattern ID)	Type: ExtDLUint Valid range: 1..5
DauxChBirth (channel-hopping reference starting point)	Type: Unsigned8
DauxSfPeriod (number of timeslots in each superframe cycle)	Type: ExtDLUint
DauxSfBirth (superframe cycle starting point)	Type: ExtDLUint Valid range: 0..127
DauxChRate (length of each slow-channel-hopping period, in number of timeslots)	Type: Unsigned8 Not transmitted and defaults to 1 when DauxOptSlowHop is FALSE
DauxChMap (channel-hopping channel map for spectrum management)	Type: Unsigned16 (LSB) Not transmitted and defaults to 0x7FFF when advChMapOv is FALSE

Table 125 summarizes the structure of the join superframe information field in the advertisement DAUX. ExtDLUint fields are shown as one octet.

Table 125 – Join superframe information structure

Number of octets	Bits							
	7	6	5	4	3	2	1	0
2	DauxTsDur (LSB)							
1	DauxChIndex							
1	DauxChBirth							
1..2	DauxSfPeriod							
1	DauxSfBirth							
0..1	DauxChRate							
0..2	DauxChMap (LSB)							

The join superframe information field is 6..10 octets.

Each subfield of advertisement join superframe information corresponds to a field in dlmo.Superframe. See 9.4.3.5.

When creating superframe number1 from the advertisement, the DL uses values from the advertisement to initialize corresponding superframe fields. Fields in the superframe number1 that do not have equivalently named fields in the advertisement default to fixed values. Table 126 shows the mapping from the advertisement's subfields to superframe number1.

Table 126 – Superframe derived from advertisement

Superframe field name	Value	Notes
* Index	1	—
TsDur	DauxTsDur	—
ChIndex	DauxChIndex	—
ChBirth	DauxChBirth	—
SFType	0	—
Priority	0	—
ChMapOv	DauxChMapOv	—
IdleUsed	0	—
SfPeriod	DauxSfPeriod	Compressed data type used in advertisement. Limits superframes used for joining to a period of approximately 300 s
SfBirth	DauxSfBirth	Compressed data type used in advertisement, consistent with SfPeriod
ChRate	DauxChRate	Compressed data type used in advertisement. Limits superframes used for joining to a slow-channel-hopping rate of approximately one hop per 2,5 s
ChMap	DauxChMap	Convert LSB to MSB
IdleTimer	null	—
rndSlots	null	—

9.3.5.2.4.2 Advertisement join links

NOTE 1 The joining process, including solicitation and advertisements and use of the information conveyed in advertisements, is described in 7.4.

There are two sets of links related to joining provided in every advertisement:

- an outbound set of links for transmitting join requests, used to initialize dlmo.Link number1; and
- an inbound set of links for receiving join responses, used to initialize dlmo.Link number2;

in addition, the advertising router may provide a set of links that a DLE can use to scan for advertisements when joining is complete, used to initialize dlmo.Link number3 when provided.

Each device that is attempting to join a subnet, upon receiving an advertisement of a D-subnet that it chooses to join, shall configure its inbound and outbound links based on the information in the received advertisement. It shall then transmit its join request to the advertising router, using those outbound links.

Links used for joining are constrained to a basic set of features. Default timeslot templates are used; see Table 165, Table 166, and Table 167.

Three types of links are identified:

– JoinTx links are used for transmitting join requests to the advertising router;
– JoinRx links are monitored while waiting for a join response;
– AdvRx links, when provided, are activated when joining is complete to passively scan for advertisements from alternative routers.

Table 127 specifies the join information field in the advertisement DAUX.

Table 127 – Join information elements

Element name	Element encoding
DauxJoinBackoff (maximum extent of backoff and retry while joining)	Type: Unsigned4
DauxJoinTimeout (join timeout)	Type: Unsigned4
DauxJoinFldXmit (indicates fields that are transmitted)	Type: Unsigned8
DauxJoinTx (JoinTx link(s))	Type: See Table 184
DauxJoinRx (JoinRx link(s))	Type: See Table 184
DauxAdvRx (Advertisement link(s))	Type: See Table 184 (or null)

The element DauxJoinFldXmit selects the link scheduling parameters used for elements DauxJoinTx, DauxJoinRx, and DauxAdvRx. DauxJoinFldXmit values 0..3 correspond to the semantics described in Table 184.

Table 128 illustrates the structure of the join information field. Fields DauxJoinTx, DauxJoinRx, and DauxAdvRx may be 1..4 octets (or null only for DauxAdvRx), depending on the configuration and selection as described in Table 184.

Table 128 – Join information structure

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	DauxJoinBackoff				DauxJoinTimeout			
1	DauxJoinFldXmit							
1..4	DauxJoinTx							
1..4	DauxJoinRx							
0..4	DauxAdvRx (may be absent)							

Depending on the alternatives selected, it would appear from Table 128 that the total size could be anything between four and fourteen octets. However, alternatives shall be selected such that the total size of the fields shown in Table 128 does not exceed ten octets.

As shown in Table 128, attributes include:

a) DauxJoinBackoff. Maximum extent of exponential backoff on joining. If a join request does not receive an ACK/NAK DPDU due to CCA channel activity detection, or a failed transmission, the DLE shall back off by selecting a uniform-random time interval in the range of 0 s to 1 s for the first retry, and use the first available JoinTx timeslot after that time. Then double the time range with each retry. The DLE may retry up to DauxJoinBackoff times and shall not retry more than DauxJoinBackoff times. At that point, the DLE should abort the attempt to send a message to the advertising router, revert to the provisioned state, and search for another advertisement.

b) DauxJoinTimeout. Guaranteed time, in s, to receive a system manager response to a join request. Expressed as an exponent, to the power of 2. For example, if DauxJoinTimeout=5, then the DLE can expect completion within 2^5 s (= 32 s). Following a timeout, the DLE should abort the attempt to join through the advertising router, revert to the provisioned state, and search for another advertisement.

c) DauxJoinFldXmit:

   – Bits 7..6: Unsigned2, describing contents of DauxJoinTx. See Table 184. Corresponds to dlmo.Link[ ].SchedType for link number1. Supported range is 0..2.

   – Bits 5..4: Unsigned2, describing contents of DauxJoinRx. See Table 184. Corresponds to dlmo.Link[ ].SchedType for link number2. Supported range is 0..2.

- Bit 3: If Bit3=1, transmit DauxAdvRx. If Bit3=0, DauxAdvRx is null and not transmitted.
- Bits 2..1: Unsigned2, describing contents of DauxAdvRx. See Table 184. Supported range is 0..2. Corresponds to dlmo.Link[ ].SchedType for link number3. If Bit3=0, Bits 2 and 1 are meaningless and shall also be 0.
- Bit 0 is reserved and shall be set to zero.

d) DauxJoinTx. The join transmission timeslot(s) in each superframe cycle, corresponding to dlmo.Link[ ].Schedule for link number1. These are the transmission opportunities in which to send join requests. Bits 7, 6 of DauxJoinFIdXmit specify the format of DauxJoinTx, as defined in Table 184.

e) DauxJoinRx, The join receive timeslot(s) in each superframe cycle, corresponding to dlmo.Link[ ].Schedule for link number2. Bits 5, 4 of DauxJoinFIdXmit specify the format of DauxJoinRx as defined in Table 184.

f) DauxAdvRx. Receive links, for scanning for additional neighbors after joining, corresponding to dlmo.Link[ ].Schedule for link number3 when provided. Bits 2, 1 of DauxJoinFIdXmit specify the format of DauxAdvRx as defined in Table 184. It is transmitted and meaningful only when DauxJoinFldXmit.Bit 3=1; otherwise its value is null and no corresponding links are created in Table 129.

Links are added to dlmo.Link[ ] based on parameters in the advertisement. Fields are set as shown in Table 129.

Table 129 – Defaults for links created from advertisements

Field name	DauxJoinTx	DauxJoinRx	DauxAdvRx (when DauxJoinFldXmit.Bit3 =1)
* Index	1	2	3
SuperframeIndex	1	1	1
Type-Transmit	1	0	0
Type-Receive	0	1	1
Type-Exponential Backoff	1	0	0
Type –Idle	0	0	1
Type-Discovery	0	0	0
Type-JoinResponse	0	0	0
Type-SelectiveAllowed	1	1	1
Template1	2	1	3
Template2	Null	Null	Null
NeighborType	1	0	0
Graph Type	0	0	0
SchedType	From advertisement DauxJoinFldXmit Bits 7..6	From advertisement DauxJoinFldXmit Bits 5..4	From advertisement DauxJoinFldXmit Bits 2..1
ChType	0	0	0
PriorityType	0	0	0
Neighbor	Address of advertising neighbor	Null	Null
GraphID	Null	Null	Null
Schedule	From advertisement DauxJoinTx	From advertisement DauxJoinRx	From advertisement DauxAdvRx
ChOffset	Null	Null	Null
Priority	Null	Null	Null

Link number3, intended to be used to scan for neighbors after joining, is configured as an idle link. Normally, idle links are enabled through a DAUX subheader as described in 9.3.5.4. In addition, when the DL changes to the join state, it shall activate link number3 for a period of time equal to the initial value of dlmo.DiscoveryAlert.Duration (default 60 s). This causes the DL to collect information into the dlmo.Candidates table and then report it to the system manager through the NeighborDiscovery alert, unless the DL is reconfigured by the system manager during the interval for a different result.

NOTE 2 GraphType in Table 129 is set to zero, indicating that the feature is not applicable in this context. See 9.4.3.7.2.

The links created from the advertisement also need entries in dlmo.Neighbor, dlmo.Graph, and dlmo.Route. These entries are automatically added by the DL at the same time as the links, with values as shown in Table 130, Table 131, and Table 132 below.

Table 130 – dlmo.Neighbor entry created from advertisements

Field name	Value
*** Index**	**Address of advertising router**
EUI64Address	Acquired from the advertising router as described in 9.1.10.1
GroupCode	0
ClockSource	2
ExtGrCnt	0
DiagLevel	1
LinkBacklog	0
ExtendGraph	Null
LinkBacklogIndex	Null
LinkBacklogDur	Null

NOTE 3 ExtendGraph in Table 130 is set to null, indicating that the feature is not applicable in this context. See 9.4.3.4.2.

Table 131 – dlmo.Graph entry created from advertisements

Field name	Value
*** Index**	**1**
PreferredBranch	0
NeighborCount	1
Queue	0
MaxLifetime	0
Neighbors	Address of advertising router

Table 132 – dlmo.Route entry created from advertisements

Field name	Value
*** Index**	**1**
Size	1
Alternative	3
ForwardLimit	16
Route	One entry: 0xA001 (Graph number1)
Selector	Null

NOTE 4 Route information in Table 132 is intended to be used as the default route after the DLE joins the D-subnet. Join messages to the neighboring proxy use source routing as described in 9.3.3.6. Sample DL headers for join messages are provided in Annex T.

DLMO updates from DAUX join information are made at two points in the DL lifecycle. First, when a DLE in the default state receives an advertisement from a provisioning mini-D-subnet (SubnetID=1), it needs to update DLMO attributes with DAUX join information in order to join the mini-D-subnet. Second, when a DLE in the provisioned state joins a target D-subnet via an advertising router, it needs to update the DLMO with DAUX join information in order to join the target D-subnet.

There are various other times when a DL might receive and process advertisements, such as when searching for multiple candidate neighbors in the provisioned state, searching for candidate neighbors in the joined state, or receiving D-subnet time updates in any state. The receipt and processing of such advertisements may trigger a join request only in the default or provisioned state, and DAUX join information in the advertisement is posted to the recipient's DLMO only when the DLE attempts to join a mini-D-subnet from the default state or attempts to join a target D-subnet from the provisioned state.

9.3.5.2.4.3 Slotted-hopping, slow-hopping and the joining process

An advertisement conveys a compressed form of a superframe definition. DauxChRate in the advertisement exactly corresponds to ChRate in the superframe, which is what distinguishes a slow-hopping superframe from a slotted-hopping one. If DauxChRate=1 the advertisement specifies a slotted-hopping superframe; when DauxChRate>1, the advertisement specifies a slow-hopping superframe.

An advertisement can specify a range of links within a superframe, per Table 184. That provides a mechanism to specify and activate a contiguous set of timeslots.

The slotted-hopping or slow-hopping specified by an advertisement DPDU is a functional subset of the slotted-hopping or slow-hopping specified by superframe and link data structures. The primary difference is that the information in the advertisement DPDU is somewhat compressed. That functional identity (for the subset) permits the information conveyed in a received advertisement DPDU to be used to initialize the superframe/link data structures in the device that is attempting the join operation.

The handling of advertisement superframes and links is described in 9.1.14. The data cross-mapping is specified in Table 126 and Table 129. The only significant difference is that, due to the compressed representation, the advertisement DPDU structure limits the slow-hopping response to 255 timeslots (typically about 2,5 s), which is noted in the description of the DPDU structure. That 2,5 s upper bound is not considered a significant limitation.

The use of slotted hopping, slow hopping and hybrid-hopping is described in 9.1.8.4.4 through 9.1.8.4.6. Figure 74 shows that slow-hopping can be combined with slotted-hopping to provide hybrid-hopping. In an advertising router configured for hybrid operation, an advertisement can instruct the joining device to use slow-hopping links, slotted-hopping links, or a combination.

For example, Tx links (to the advertising router) might use slow hopping while Rx links (from the advertising router) might use slotted hopping. Such a configuration is reasonable for a router that is configured as an active scanning host (see 9.1.13.3), where slow-hopping links can perform double-duty as a vehicle for listening for solicitation DPDUs and other Data DPDUs.

The slotted-hopping or slow-hopping or hybrid-hopping pattern is defined by superframe attributes in Table 174. The mapping is shown in Table 126. The timing of the links within the slotted-hopping or slow-hopping or hybrid-hopping superframe is defined in Table 181 and Table 184. Table 127 shows the mapping to Table 181 while referring specifically to Table 184.

In a hybrid configuration, such as in Figure 74, slow-hopping links may be limited to a particular range of timeslots. That is the intended use of the "range" in Table 184. It is also possible to designate specific timeslots (links) within a slow-hopping period, not just a range of timeslots, thus providing more flexibility for the designer of an actual deployed WISN.

9.3.5.2.4.4 Integrity check

DPDUs that embed a DAUX include the IEEE 16-bit ITU-T CRC (FCS) as an integrity check on the overall MPDU, plus a DMIC for authentication as a further integrity check. However, that DMIC cannot be authenticated by a receiving DLE without a shared sense of time, a shared security key, and knowledge of the sending DLE's EUI64Address, which are not available to all DLEs that may overhear the DPDU and derive time from its DAUX subheader.

NOTE 1 Time within the DAUX is usable as a shared sense of time for authentication of the DPDU that contains the DAUX.

This standard permits a DLE to view and use the DAUX even if it cannot authenticate the overall DPDU. It was deemed insufficient to rely on the IEEE 802.15.4 FCS as the only integrity check for advertisements. For this reason, an additional 16-bit integrity check is included within the DAUX, covering only the contents of the DAUX itself.

The DAUX integrity check is similar to the UDP checksum described in IETF RFC 768. The checksum is the 16-bit ones complement of the ones' complement sum of the octets that comprise the DAUX subheader, excluding the integrity check itself, padded with zero octets at the end (if necessary) to make a multiple of two octets. Octet ordering is as transmitted. If the computed checksum is zero, it is transmitted as all ones. An all-zero transmitted checksum value means that the creator of the DPDU generated no checksum.

Transmission order of the integrity check is MSB, i.e., with the first octet reflecting bit operations on odd-numbered octets, with the octet count starting at 1, in the DAUX subheader (octets 1, 3, 5, etc.) and the second octet from even-numbered octets in the DAUX subheader (octets 2, 4, 6, etc.).

NOTE 2 The use of a secret D-subnet security key for advertisements enables those advertisements to be trusted after the DLE has joined the D-subnet. Advertisements are usable for periodic surveys of neighboring routers, or for periodic time updates by DLEs with low-duty cycles.

The advertisement provides only the DL16Address of the advertising router. However, an EUI64Address is needed for subsequent exchange of DPDUs with that router. As described in 9.1.14.2, the responding DLE shall acquire the EUI64Address from the advertising DLE.

9.3.5.2.5 Configuring advertisements

NOTE The joining process, including solicitation and advertisements and use of the information conveyed in advertisements, is described in 7.4.

The timing of advertisements is determined by the structure of the advertising DLE's superframes and links. Any link may include an advertisement flag, which indicates that the advertisement is included in the DAUX.

An index value in the attribute dlmo.AdvSuperframe (see Table 141) selects a superframe in dlmo.Superframe that shall be used as a reference to build the advertisement. The reference superframe is configured by the system manager by establishing a superframe, which may be idle, and referring to its index in the dlmo.AdvSuperframe attribute. The reference superframe shall not use features that cannot be represented in the join superframe information field in Table 124.

A zero value in dlmo.AdvSuperframe is the default, and indicates that the advertisement has not been configured.

Link information is placed in the dlmo.AdvJoinInfo attribute, in exactly the format in which it is transmitted in the advertisement in the position corresponding to Table 128. In this way, the new DLEs JoinTx, Join Rx, and AdvRx links are specified.

The system manager configures superframes in the advertising router that match those specified in the advertisement DPDU. At the time of JoinTx links, the advertising router shall be configured with links to receive join DPDUs. At the time of JoinRx links, the advertising router shall be configured with links where JoinResponse=1 (see Table 182).

9.3.5.3 Solicitation auxiliary subheader

9.3.5.3.1 General

NOTE The joining process, including solicitation and advertisements and use of the information conveyed in advertisements, is described in 7.4.

A solicitation is a request for an advertisement to be transmitted by an active scanning host in range, on the same channel as the solicitation itself (see 9.1.13.3).

Attributes within a solicitation DAUX can be grouped logically as:

- solicitation header; and
- D-subnet ID.

The solicitation does not have a reliable sense of time, nor does it necessarily have a secret security key. Therefore, to allow the receiver of the solicitation to decode its DMIC, a solicitation's DMIC shall be built using a security key of K_global and a nominal TAI time of zero. This allows for consistent processing, and provides a strong integrity check for the DPDU. No additional integrity check is included in the solicitation's DAUX.

A solicitation's MHR (IEEE MAC header) shall not provide a source or destination address, and it shall not specify a D-subnet. See 9.1.5.

9.3.5.3.2 Solicitation fields

Table 133 specifies the solicitation header in the solicitation DAUX.

Table 133 – Solicitation header subfields

Subfield name	Subfield encoding
DauxType	Type: Unsigned3 1= solicitation DAUX
DauxSubnetInclude (indicates whether to transmit DauxSubnetID in solicitation)	Type: Unsigned1
Reserved	Type: Unsigned4=0

The solicitation header is 1 octet. As shown in Table 133, elements include:

- DauxType. Set to 1 to indicate a solicitation DAUX.
- DauxSubnetInclude. Indicates whether to transmit the DauxSubnetID field in the solicitation. If DauxSubnetInclude=0, the DauxSubnetID field is not transmitted, and the receiver (active scanning host) shall use the default value of DauxSubnetID=0 in filtering.

Table 134 illustrates the structure of the solicitation header.

Table 134 – Solicitation header structure

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	DauxType=1			DauxSubnetInclude	Reserved=0			

Table 135 specifies the other fields in the solicitation DAUX.

Table 135 – Solicitation DAUX fields

Field name	Field encoding
DauxSubnetID (specifies the SubnetID)	Type: Unsigned16 (LSB)

DauxSubnetID transmits a D-subnet ID that can be used as a filter by the receiver, based on the receiver's dlmo.SolicFilter attribute. When DauxSubNetInclude=0, DauxSubnetID defaults to 0x0000 and is not transmitted.

Table 136 summarizes the structure of the solicitation DAUX.

Table 136 – Solicitation DAUX structure

Number of octets	bits							
	7	6	5	4	3	2	1	0
1	Solicitation header (see Table 134)							
0, 2	DauxSubnetID (LSB)							

9.3.5.3.3 Configuring solicitations

Due to regulatory and safety requirements, some applications cannot tolerate DLEs that transmit DPDUs while they are idle or in transit. Therefore, the default in this standard does not include solicitations in its configuration. The intent is that DLEs will be configured with solicitations, as appropriate, when they are provisioned or subsequently in the lifecycle.

The timing of solicitations is determined by the structure of the DLE's superframes and links. Transmission of a solicitation is triggered by a dlmo.Link[ ].Discovery field set to a value of 3.

When a solicitation is transmitted, the contents of dlmo.SolicTemplate shall be copied verbatim into the DAUX subheader. If the size of SolicTemplate is zero, this shall be interpreted as a configuration error and the link shall be ignored.

To support regulatory and safety requirements, solicitations can be enabled and disabled on a timed basis by the system manager through the attributes dlmo.RadioSilence, dlmo.RadioSleep, and dlmo.Superframe[ ].IdleTimer.

9.3.5.4 Activate link auxiliary subheader

9.3.5.4.1 General

The activate link DAUX provides a mechanism that enables a transaction initiator to activate idle timeslots for a short period of time in order to efficiently forward a backlog of messages that have accumulated in a transaction initiator's message queue. These idle links, when so configured by the system manager, are activated by the router in response to a burst of messages flowing through a DL toward a particular neighbor.

The system manager configures:

- An idle transmit link on the transmission side, addressed to a particular neighbor or group of neighbors, with a particular link index and schedule.
- An idle receiver link on the reception side, with the same link index and schedule.
- A set of parameters in the neighbor table, indicating
  - the link index (LinkBacklogIndex),
  - the size of the backlog that should trigger link activation (LinkBacklog), and
  - the duration of link activation (LinkBacklogDur).

  See 9.4.3.4.2 for the definition of these parameters.

When the transaction initiator detects that there are LinkBacklog Data DPDUs on its message queue that can be forwarded to the Data DPDU's destination address, the transaction initiator should use the activate link auxiliary subheader to activate the idle receive link through the activate link auxiliary subheader. When the transaction originator receives an ACK/NAK DPDU for the Data DPDU, that implies that the message has been processed and that the receive side of the idle link has been activated. The transaction initiator should then activate the transmit side of the idle link for the designated number of communication opportunities.

The activated transmit link might be addressed to a group of neighbors, however, the receive side of the activated link occurs on only one neighbor. Therefore, only Data DPDUs addressed to that neighbor should be considered as candidates for the activated link.

The activate link DAUX provides a link index and a number of communication opportunities that are used to activate an idle link by the receiver of the Data DPDU. It has the result of immediately activating an idle link for reception, for the number of communication opportunities indicated by DauxActivateDur. The transaction initiator of the activate link message is, in essence, informing the receiver that queued messages will be following that will be sent during communication opportunities (timeslots) associated with a particular receive link.

Activation of idle links is triggered on the transaction initiator side by multiple queued Data DPDUs that can be routed to the receiver. See 9.4.3.4.2 for a description of the transmit side of the activate link message (LinkBacklogIndex, LinkBacklogDur).

9.3.5.4.2 Fields

Table 137 summarizes the activate link DAUX.

Table 137 – Activate link DAUX fields

Field name	Field encoding
DauxType	Type: Unsigned3 2=Link activation DAUX
Reserved (octet alignment)	Type: Unsigned5=0
DauxLink_ID (identifier for link)	Type: ExtDLUint
DauxActivateDur (number of communication opportunities (timeslots) to activate the link, link occurrences)	Type: Unsigned8

The link is activated for the number of occurrences of the link, whether the link is used or not. For example, the link occurrence is counted even if it is not used because of a higher priority link in the same timeslot. The link activation period begins with the next full timeslot after the link activation DAUX is received. See 9.4.3.4.2.

Table 138 illustrates the structure of the activate link DAUX field.

Table 138 – Activate link DAUX structure

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	DauxType			Reserved=0				
1 or 2octets	DauxLinkID							
1t	DauxActivateDur							

9.3.5.5 Signal quality auxiliary subheader

9.3.5.5.1 General

The signal quality DAUX reports the quality of the received signal in an ACK/NAK DPDU, to support collection of round-trip signal quality diagnostics. Two octets are reported, one for signal strength (RSSI) and one for signal quality (RSQI). RSSI and RSQI are described in 9.1.15.2.

9.3.5.5.2 Fields

Table 139 summarizes the report received signal quality DAUX.

Table 139 – Report received signal quality DAUX fields

Field name	Field encoding
DauxType	Type: Unsigned3 3=Signal quality DAUX
Reserved (octet alignment)	Type: Bit5=0
DauxRSSI (RSSI)	Type: Integer8
DauxRSQI (RSQI)	Type: Unsigned8

Table 140 illustrates the structure of the report received signal quality DAUX.

Table 140 – Report received signal quality DAUX structure

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	DauxType			Reserved=0				
1	DauxRSSI							
1	DauxRSQI							

9.4 DL management information base

9.4.1 General

For information on the general handling of standard management objects in the DL, see 9.1.11.

9.4.2 DL management object attributes

9.4.2.1 General

Table 141 summarizes the DL management object (DLMO) attributes. OctetStrings with a size of zero are referred to as null.

Table 141 – DLMO attributes *(1 of 7)*

Standard object type name: DL management object (DLMO)				
Standard object type identifier: 124				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
ActScanHostFract	1	DLE's behavior as an active scanning host	Type: Unsigned8	See 9.4.2.2
			Classification: Static	
			Accessibility: Read/write	
			Default value: 0	
AdvJoinInfo	2	Join information to be placed in advertisement	Type: OctetString	See 9.4.2.3
			Classification: Static	
			Accessibility: Read/write	
			Default value: Null	
AdvSuperframe	3	Superframe reference for advertisement	Type: Unsigned16	See 9.4.2.3
			Classification: Static	
			Accessibility: Read/write	
			Default value: 0	
			Valid range: 0..32 767	
SubnetID	4	Identifier of the D-subnet that the DLE has joined or is attempting to join	Type: Unsigned16	See 9.4.2.4
			Classification: Dynamic	
			Accessibility: Read only	
			Default value: 0	
SolicTemplate	5	Template of DAUX subheader used for solicitations	Type: OctetString	See 9.4.2.5
			Classification: Static	
			Accessibility: Read/write	
			Default value: Null	

Standard object type name: DL management object (DLMO)				
Standard object type identifier: 124				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
AdvFilter	6	Filter used on incoming advertisements during neighbor discovery	Type: OctetString See Table 142	See 9.4.2.20
			Classification: Static	
			Accessibility: Read/write	
			Default value: See 9.4.2.20	
SolicFilter	7	Filter used on incoming solicitations	Type: OctetString See Table 142	See 9.4.2.20
			Classification: Static	
			Accessibility: Read/write	
			Default value: See 9.4.2.20	
TaiTime	8	TAI time for DLE	Type: TAINetworkTime	Units: 2^{-16} s See 9.4.2.6
			Classification: Static	
			Accessibility: Read only	
TaiAdjust	9	Adjust TaiTime at an instant that is scheduled by the system manager	Type: OctetString See Table 143	See 9.4.2.21
			Classification: Dynamic	
			Accessibility: Read/write	
			Default value: Null	

Table 141 *(2 of 7)*

Standard object type name: DL management object (DLMO)				
Standard object type identifier: 124				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
MaxBackoffExp	10	Maximum backoff exponent for retries	Type: Unsigned8	See 9.4.2.7
			Classification: Static	
			Accessibility: Read/write	
			Default value: 5	
			Valid range: 3..8	
MaxDsduSize	11	Maximum octets that can be accommodated in a single DSDU	Type: Unsigned8	See 9.4.2.8
			Classification: Static	
			Accessibility: Read/write	
			Default value: 96	
			Valid range: 76..96	
MaxLifetime	12	Maximum Data DPDU lifetime	Type: Unsigned16	Units: 0,25 s See 9.4.2.9
			Classification: Static	
			Accessibility: Read/write	
			Default value: 120 (30 s)	
			Valid range: 8..1 920 (2 s..480 s)	
NackBackoffDur	13	Duration of backoff after receiving a NAK	Type: Unsigned16	Units: 0,25 s See 9.4.2.10
			Classification: Static	
			Accessibility: Read/write	
			Default value: 60 (15 s)	
			Valid range: 8..1 920 (2 s..480 s)	
LinkPriorityXmit	14	Default priority for transmit links	Type: Unsigned8	See 9.4.2.11
			Classification: Static	
			Accessibility: Read/write	
			Default value: 8	
			Valid range: 0..15	
LinkPriorityRcv	15	Default priority for receive links	Type: Unsigned8	See 9.4.2.11
			Classification: Static	
			Accessibility: Read/write	
			Default value: 0	
			Valid range: 0..15	
EnergyDesign	16	DLE's energy capacity as designed	Type: OctetString	See 9.4.2.22
			Classification: Constant	
			Accessibility: Read only	
			Default value: See 9.4.2.22	

Table 141 *(3 of 7)*

Standard object type name: DL management object (DLMO)				
Standard object type identifier: 124				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
EnergyLeft	17	Remaining energy for DLE	Type: Integer16 Classification: Dynamic Accessibility: Read only	See 9.1.17
DeviceCapability	18	Device capabilities	Type: OctetString See Table 147 Classification: Constant Accessibility: Read only Default value: See 9.4.2.23	See 9.4.2.23
IdleChannels	19	Radio channels that shall be idle	Type: Unsigned16 Classification: Static Accessibility: Read/write Default value: 0	See 9.4.2.12
ClockExpire	20	Clock expiration deadline	Type: Unsigned16(MSB) Classification: Static Accessibility: Read/write Default value: See description	Units 1s See 9.4.2.13
ClockStale	21	DL clock source timeout	Type: Unsigned16 Classification: Static Accessibility: Read/write Default value: See description Valid range: 5..300	Units: 1 s See 9.4.2.14
RadioSilence	22	Radio silence timeout	Type: Unsigned32 Classification: Static Accessibility: Read/write Default value: 600 Valid range: Limited to 0..600 for radio silence profile; otherwise $0..2^{32}-1$	Units: 1s See 9.4.2.16 and 9.1.15.4
RadioSleep	23	Radio sleep period. Note: DLE's radio will be disabled when this attribute is set	Type: Unsigned32 Classification: Dynamic Accessibility: Read/write Default value: 0	Units: 1s See 9.4.2.17
RadioTransmitPower	24	Radios maximum transmit power level	Type: Integer8 Classification: Static Accessibility: Read/write Default value: See text Valid range: -20..36	Units: dBm See 9.4.2.18

Table 141 *(4 of 7)*

Standard object type name: DL management object (DLMO)				
Standard object type identifier: 124				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
CountryCode	25	Information about the device's regulatory environment	Type: Unsigned16	See 9.4.2.19, 9.1.15.6 and Annex V
			Classification: Static	
			Accessibility: Read/write	
			Default value: 0x3C00	
Candidates	26	A list of candidate neighbors discovered by the DLE	Type: OctetString See Table 151	See 9.4.2.24
			Classification: Dynamic	
			Accessibility: Read/write	
			Default value: Null	
DiscoveryAlert	27	Control of NeighborDiscovery alert	Type: OctetString	See 9.4.2.24
			Classification: Dynamic	
			Accessibility: Read/write	
			Default value: See 9.4.2.25	
			Valid range: See 9.4.2.24	
SmoothFactors	28	Smoothing factors for diagnostics	Type: OctetString See Table 153	See 9.4.2.25
			Classification: Static	
			Accessibility: Read/write	
			Default value: See Table 153	
			Valid range: See Table 153	
QueuePriority	29	Queue buffer capacity for specified priority level	Type: OctetString See Table 155	See 9.4.2.26
			Classification: Static	
			Accessibility: Read/write	
			Default value: N=0	
Ch	30	Channel-hopping patterns	Type: OctetString (indexed) See Table 159	See 9.4.3.2
			Classification: Static	
			Accessibility: Read/write	
			Default value: See 9.4.3.2	
			Valid range: See 9.4.3.2	
ChMeta	31	Metadata for Ch attribute	Type: Metadata_attribute	See 9.4.3.2 [a]
			Classification: Static	
			Accessibility: Read only	

Table 141 *(5 of 7)*

Standard object type name: DL management object (DLMO)				
Standard object type identifier: 124				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
TsTemplate	32	Timeslot templates	Type: OctetString (indexed) See Table 161 and Table 163	See 9.4.3.3
			Classification: Static	
			Accessibility: Read/write	
			Default value: See 9.4.3.3	
			Valid range: See 9.4.3.3	
TsTemplateMeta	33	Metadata for TsTemplate attribute	Type: Metadata_attribute	See 9.4.3.3 [a]
			Classification: Static	
			Accessibility: Read only	
Neighbor	34	Neighbors	Type: OctetString (indexed) See Table 168	See 9.4.3.4
			Classification: Static	
			Accessibility: Read/write	
			Default value: Empty	
			Valid range: See 9.4.3.4	
NeighborDiagReset	35	Used to update DiagLevel field within Neighbor attribute	Type: OctetString (indexed) See Table 172	See 9.4.3.4.3
			Classification: Static	
			Accessibility: Read/write	
			Valid range: See 9.4.3.4.3	
NeighborMeta	36	Metadata for Neighbor attribute	Type: Metadata_attribute	See 9.4.3.4 [a]
			Classification: Static	
			Accessibility: Read only	
Superframe	37	Superframes; structures and activation	Type: OctetString (indexed) See Table 175	See 9.4.3.5
			Classification: Dynamic	
			Accessibility: Read/write	
			Default value: Empty	
			Valid range: See 9.4.3.5	
SuperframeIdle	38	Used to update idle fields within Superframe attribute	Type: OctetString (indexed) See Table 177	See 9.4.3.5.3
			Classification: Dynamic	
			Accessibility: Read/write	
SuperframeMeta	39	Metadata for Superframe attribute	Type: Metadata_attribute	See 9.4.3.5 [a]
			Classification: Static	
			Accessibility: Read only	

Table 141 *(6 of 7)*

Standard object type name: DL management object (DLMO)				
Standard object type identifier: 124				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Graph	40	Graphs	Type: OctetString (indexed) See Table 178 Classification: Static Accessibility: Read/write Default value: Empty	See 9.4.3.6
GraphMeta	41	Metadata for Graph attribute	Type: Metadata_attribute Classification: Static Accessibility: Read only	See 9.4.3.6 [a]
Link	42	Links	Type: OctetString (indexed) See Table 180 Classification: Static Accessibility: Read/write Default value: Empty	See 9.4.3.7
LinkMeta	43	Metadata for Link attribute	Type: Metadata_attribute Classification: Static Accessibility: Read only	See 9.4.3.7 [a]
Route	44	Routes	Type: OctetString (indexed) See Table 185 Classification: Static Accessibility: Read/write Default value: Empty	See 9.4.3.8
RouteMeta	45	Metadata for Route attribute	Type: Metadata_attribute Classification: Static Accessibility: Read only	See 9.4.3.8 [a]
NeighborDiag	46	Neighbor link diagnostics	Type: OctetString (indexed) See Table 187 Classification: Dynamic Accessibility: Read only Default value: Empty	See 9.4.3.9
DiagMeta	47	Metadata for NeighborDiag attribute	Type: Metadata_attribute Classification: Static Accessibility: Read only	See 9.4.3.9 [a]
ChannelDiag	48	Per-channel diagnostics for spectrum management	Type: OctetString Classification: Dynamic Accessibility: Read only Default value: See 9.4.2.27	See 9.4.2.27

Table 141 *(7 of 7)*

Standard object type name: DL management object (DLMO)				
Standard object type identifier: 124				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
AlertPolicy	49	Report diagnostics if connectivity problems are detected between regular HRCO reports	Type: OctetString	See 9.6.1
			Classification: Static	
			Accessibility: Read/write	
			Default value: See 9.6.1	
DLTimeout	50	DLE may reasonably reset to provisioned state if it doesn't receive a time update in this time interval	Type: Unsigned16	See 9.4.2.15
			Classification: Static	
			Accessibility: Read/write	
			Default value: See description	
			Valid range: > 0	
[a] Metadata containing a count of the number of entries in the table and capacity (the total number of rows allowed) for the table; see 6.2.6.3 for details about this data type.				

9.4.2.2 dlmo.ActScanHostFract

dlmo.ActScanHostFract configures the DLE's behavior as an active scanning host, as specified in 9.1.13.3. The setting indicates the fraction of time that the DLE should respond when it receives an active scanning solicitation. The default of 0 indicates that the DLE is not configured as an active scanning host.

9.4.2.3 dlmo.AdvJoinInfo and dlmo.AdvSuperframe

dlmo.AdvJoinInfo and dlmo.AdvSuperframe configure the contents of an advertisement's DAUX subheader. Their meaning is described in 9.3.5.2.5.

9.4.2.4 dlmo.SubnetID

dlmo.SubnetID is the identifier for the single D-subnet that the DLE is currently using or attempting to join. The DL management SAPs handle only one active D-subnet at a time. If a given device is participating in multiple D-subnets concurrently, this may be modeled as multiple instances of the DLE. dlmo.SubnetID=0 shall never be used as a D-subnet ID; its use indicates that the DLE is not participating in a D-subnet. dlmo.SubnetID=1 shall be used exclusively to identify provisioning D-subnets. The system manager doesn't set the DLE's SubnetID directly; rather, it is set by the DLE itself in the process of discovering and joining the D-subnet. See 9.1.10.2.

NOTE In the IEEE 802.15.4 design, SubnetID is usable as a filter for incoming MPDUs. As discussed in 9.1.10.2, a DMIC provides an additional, stronger and more reliable filter once the DLE has joined the D-subnet.

9.4.2.5 dlmo.SolicTemplate

dlmo.SolicTemplate is a template for the DAUX subheader in a solicitation. When a solicitation is transmitted, the exact data in this OctetString (without a prepended explicit size) shall be used as the DAUX subheader. It is null (zero size) by default. See 9.3.5.3.

9.4.2.6 dlmo.TaiTime

dlmo.TaiTime, when read by the DMAP, is reported as the DLE's best estimate of DL time at that instant. See 12.22.4.2 for encoding of TAINetworkTime.

NOTE The dlmo.TaiTime attribute is described as a read-only attribute, where time is acquired by the DLE from its neighbors and provided as a service to other layers through the DMAP. This style of specification is not intended to preclude implementations, such as on DLEs that are clock masters, where time is provided to the DLE from an alternative source.

9.4.2.7 dlmo.MaxBackoffExp

dlmo.MaxBackoffExp is the maximum backoff exponent for retries; see 9.1.8.2 for a discussion of exponential backoff.

9.4.2.8 dlmo.MaxDsduSize

dlmo.MaxDsduSize is the maximum number of octets that can be accommodated in a single DSDU. This is used by the NL to make fragmentation decisions. Its default value of 96 allows for the following constraints:

- A single EUI64Address in the MHR. See 9.3.3.2.
- A one-octet Crypto Key Identifier and a slow-channel-hopping-offset in the DMXHR. See 9.3.3.4.
- A single compressed route in DROUT (i.e. no source routing beyond the single hop case). See 9.3.3.6.
- No DAUX, so that a fragmented DSDU cannot be combined with an advertisement, with the exception of the link activation DAUX when 16-bit addressing is used.
- A DMIC-32, not DMIC-64 or DMIC-128.

NOTE MaxDsduSize was calculated as follows: 15 octets for the MHR (see 9.3.3.2); 1 octet for the DHR (see 9.3.3.3); 3 octets for the DMXHR (see 9.3.3.4); 0 octets for the DAUX (see 9.3.3.5); 2 octets for the DROUT (see 9.3.3.6); 4 octets for DADDR (see 9.3.3.7); 4 octets for the DMIC; and 2 octets for the FCS. This total of 31 octets is subtracted from the PhSDU capacity of 127 octets, to arrive at a MaxDsduSize of 96 octets.

The system manager shall reduce the value of dlmo.MaxDsduSize as needed if additional constraints apply to a particular configuration.

9.4.2.9 dlmo.MaxLifetime

dlmo.MaxLifetime is the maximum duration, in units of 0,25 s, for which a Data DPDU is to be held in the message queue of a single DLE before it shall be discarded. dlmo.MaxLifetime can be overridden by dlmo.Graph[ ].MaxLifetime (see 9.4.3.6).

9.4.2.10 dlmo.NackBackoffDur

dlmo.NackBackoffDur is the duration of the backoff, in units of 0,25 s, after receiving a NAK (see 9.1.9.4.4).

9.4.2.11 dlmo.LinkPriorityXmit and dlmo.LinkPriorityRcv

dlmo.LinkPriorityXmit and dlmo.LinkPriorityRcv are the default priorities to be used when selecting links. If no priority is specified in dlmo.Link[ ].Priority, use these priorities. For T/R links, use dlmo.LinkPriorityRcv as the priority for the receive side of the link. Link priorities are functionally described in 9.1.8.5.

9.4.2.12 dlmo.IdleChannels

dlmo.IdleChannels provides a list of channels that shall be idle, as a quick way for the system manager to block the usage of certain channels on a particular DLE without requiring a coordinated change of channel-hopping schedules. A link occurring on any of the channels designated as idle (value 1) by dlmo.ActiveChannels shall be treated as idle. Values of 1 in dlmo.IdleChannels shall not cause hop sequences to be shortened, but rather leaves the hop sequences intact and simply causes all links on designated channels to be treated as idle. (Shortening of the hop sequences themselves is accomplished through a different attribute, dlmo.Superframe[ ].ChMap.) Bit positions 0..15 correspond to channels 0..15. A bit value of 1

indicates that links using the channel shall be treated as idle. dlmo.IdleChannels is complimentary with dlmo.DeviceCapability.ChannelMap; in the operation of the DLE, the two are logically combined as follows, resulting in a set of channels that are treated as active by the DLE:

ActiveChannels = ((NOT dlmo.IdleChannels) AND (dlmo.DeviceCapability.ChannelMap))

9.4.2.13 dlmo.ClockExpire

dlmo.ClockExpire is the maximum number of seconds that the DLE can safely operate without a clock update. The default is (1 000 s/DeviceCapability.ClockStability), which is intended to keep the DLE synchronized to within 1 ms during the joining process and thereafter when participating in a D-subnet that provides only slotted-hopping. In other cases, the needed value scales linearly with the needed tighter or looser clock accuracy. See 9.1.9.2.2.

NOTE A device that requires use of slow-hopping is likely to have a longer ClockExpire duration than the above default.

9.4.2.14 dlmo.ClockStale

dlmo.ClockStale determines when the DLE should start accepting time updates from secondary DL clock sources. For example, if dlmo.ClockTimeout is set to the default of 45 s, then a DL clock recipient should not accept clock updates from a secondary DL clock source until at least 45 s has elapsed since it last received a clock update from a primary DL clock source. The default value is 0,5 × ClockExpire. See 9.1.9.2.3 for more information.

9.4.2.15 dlmo.ClockTimeout

dlmo.ClockTimeout is the maximum number of seconds that the DLE can reasonably operate in a D-subnet before resetting itself to the provisioned state. The default value is 2,0 × ClockExpire. See 9.1.9.2.2.

9.4.2.16 dlmo.RadioSilence

dlmo.RadioSilence designates when a DLE shall disable its transmitter after losing its D-subnet connection. See 9.1.15.4 for more information.

9.4.2.17 dlmo.RadioSleep

dlmo.RadioSleep is used to disable the DLE's radio for a period of time. See 9.1.15.4 for more information. Activation of this attribute shall be slightly delayed to allow for transmitting an application layer acknowledgment of the DMAP TPDU that causes the attribute to be set.

9.4.2.18 dlmo.RadioTransmitPower

dlmo.RadioTransmitPower is used to control the DLE's radio transmit power level, in dBm EIRP. It defaults to the device's maximum permitted transmit power level under the regulatory regime specified by dlmo.CountryCode (9.1.15.6), and is reported during the joining process through dlmo.DeviceCapability.RadioTransmitPower. See 9.1.15.5.

9.4.2.19 dlmo.CountryCode

dlmo.CountryCode provides constraints on a device based on the applicable regulatory regime. When set during DLE provisioning, use of the supported content-locking functionality can constrain operation of the device until it is next provisioned (e.g., perhaps after repair and deployment to a different regulatory jurisdiction). See 9.1.15.6 and Annex V.

9.4.2.20 Subnet filters

A D-subnet filter attribute is a string of 4 octets that specifies how a DLE shall filter incoming advertisements or solicitations. AdvFilter is used to filter incoming advertisements, and SolicFilter is used to filter incoming solicitations.

AdvFilter and SolicFilter each include two fields, a 16-bit BitMask field and a 16-bit TargetID field. Table 142 shows the structure of each D-subnet filter.

Table 142 – D-subnet filter octets

Number of octets	Bits							
	7	6	5	4	3	2	1	0
2	BitMask							
2	TargetID							

Unlike most DLMO attributes, D-subnet filters use LSB octet ordering conventions. This reflects their use, which is to perform bit comparison operations with DPDU header elements that are transmitted in LSB order.

When a DLE receives an advertisement, it shall check the incoming DPDU's SubnetID. The advertisement shall be ignored unless:

(DPDU.SubnetID AND AdvFilter.BitMask) equals (AdvFilter.TargetID AND AdvFilter.BitMask)

AdvFilter.BitMask shall default to 0xFFFF, and AdvFilter.TargetID shall default to 0x0001, with the result that an unprovisioned DLE in the default state will filter all advertisements except those received from a provisioning D-subnet with SubnetID=1.

When a DLE receives a solicitation, it shall check the incoming DPDU's SubnetID. The solicitation shall be ignored unless:

(DPDU.DauxSubnetID AND SolicFilter.BitMask) ==
(SolicFilter.TargetID AND SolicFilter.BitMask)

SolicFilter.BitMask shall default to 0x0000, with the result that solicitations are not filtered by default.

9.4.2.21 Time adjustments

The dlmo.TaiAdjust attribute includes fields that are used to adjust dlmo.TaiTime at an instant that is scheduled by the system manager. This attribute is normally null, unless a time correction is pending. Its use is described in 9.1.9.3.6. The OctetString comprises a series of fields that are described in Table 143.

Table 143 – dlmo.TaiAdjust OctetString fields

Field name	Field encoding
TaiCorrection (indicates the magnitude and direction of a TAI clock correction)	Type: Integer32 Units: 2^{-15} s
TaiTimeToApply (indicates the time at which the correction shall be applied)	Type: TAITimeRounded Units: 1 s

NOTE The TaiCorrection unit of 2^{-15} s was chosen to match the 32 KiHz very-precise very-low-power "watch" crystals commonly used for the continuous clock hardware of WISN devices.

Table 144 illustrates the structure of the OctetString.

Table 144 – dlmo.TaiAdjust OctetString structure

Number of octets	Bits							
	7	6	5	4	3	2	1	0
4	TaiCorrection							
4	TaiTimeToApply							

9.4.2.22 DLE energy capacity

The dlmo.EnergyDesign attribute, as shown in Table 145, includes various elements that indicate the energy capacity of the device. The fields within this attribute are described in 9.1.17.

Table 145 – dlmo.EnergyDesign OctetString fields

Field name	Field encoding
EnergyLife (DLE energy life by design; positive for months, negative for days)	Type: Integer16 (constant)
ListenRate (DLE's energy capacity to operate its receiver, in seconds per hour)	Type: ExtDLUint (constant)
TransmitRate (DLE's energy capacity to transmit DPDUs, in DPDUs per minute	Type: ExtDLUint (constant)
AdvRate (DLE's energy capacity to transmit advertisements, in DPDUs per minute)	Type: ExtDLUint (constant)

Table 146 illustrates the structure of the OctetString.

Table 146 – dlmo.EnergyDesign OctetString structure

Number of octets	Bits							
	7	6	5	4	3	2	1	0
2	EnergyLife							
1..2	ListenRate							
1..2	TransmitRate							
1..2	AdvRate							

9.4.2.23 DLMO device capabilities

The dlmo.DeviceCapability attribute includes various elements that indicate the capabilities of the device. This is a read-only attribute, most of whose component values do not change during normal operation. (They may be changed due to remote system management, including by the download of new firmware.) dlmo.DeviceCapability shall be reported to the system manager as part of the joining process.

The OctetString comprises a series of fields that are described in Table 147. Some of these fields, listed as static, do not change during operation and are reported only on joining. Other fields, listed as dynamic, may change during operation and are also available after joining through identically-named DLMO attributes.

Table 147 – dlmo.DeviceCapability OctetString fields

Field name	Field encoding
QueueCapacity (capacity of the queue that is available for forwarding operations)	Type: ExtDLUint (constant)
ClockStability (nominal clock stability of this device, as a multiple of 1×10^{-6})	Type: Unsigned8 (constant)
ChannelMap (map of radio channels supported by the device)	Type: Unsigned16 (constant)
DLE_roles (DLE roles supported by the DLE)	Type: BooleanArray8 (constant)
EnergyDesign (copy of attribute dlmo.EnergyDesign)	Type: OctetString (constant)
EnergyLeft (copy of attribute dlmo.EnergyLeft)	Type: Integer16
Ack_Turnaround (see Table 161) [a]	Type: ExtDLUint (constant)
NeighborDiagCapacity (memory capacity for dlmo.NeighborDiag)	Type: ExtDLUint (constant)
RadioTransmitPower (copy of attribute RadioTransmitPower, see 9.1.15.5)	Type: Integer8
SupportedCCAmodes (bitmap description of CCA modes supported by the device)	Type: BooleanArray8 (constant)
ConstructionOptions (i.e., optional features supported by DLE)	Type: BooleanArray8 (constant)
[a] This is the greater of the time required to switch from transmit to receive or from receive to transmit, both of which occur (in different DLEs) at the end of a Data DPDU before the following ACK/NAK DPDU	

Table 148 illustrates the structure of the OctetString. Size of EnergyDesign includes a one-octet size field prepended within the unspecified-length OctetString.

Table 148 – dlmo.DeviceCapability OctetString structure

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1..2	QueueCapacity							
1	ClockStability							
2	ChannelMap							
1	DLERoles							
6..9	EnergyDesign (OctetString size, plus 5..8 octet content)							
2	EnergyLeft							
1..2	AckTurnaround							
1..2	NeighborDiagCapacity							
1	RadioTransmitPower							
1	SupportedCCAmodes							
1	ConstructionOptions							

Fields include:

- dlmo.DeviceCapability.QueueCapacity is the number of buffers in the queue that are available for forwarding operations. This capacity shall not include internal buffers that may be used for messages that flow through the NLE. This value shall be based on the worst case, wherein all DPDUs are of maximum size. In operation, the actual queue capacity may be larger than this reported value. See 9.1.8.5.
- dlmo.DeviceCapability.ClockStability is the nominal short-term clock stability of the device, as a multiple of 1×10^{-6}, in the absence of a time correction from the D-subnet. See 9.1.9.2.2.

- dlmo.DeviceCapability.ChannelMap is a list of channels that the device can legally support in the device's regulatory domain (as determined by dlmo.CountryCode, 9.1.15.6), where a value of zero indicates that the device is not permitted to use the channel. Bit positions 0..15 correspond to channels 0..15 of this standard, which in turn correspond to 2,4 GHz DSSS channels 11..26 of IEEE 802.15.4. If the DLE is configured with links that refer to such blocked channels, the DLE shall treat those links as idle. See 9.1.7.2.3, 9.1.15.6, 9.4.2.19 and Annex V.
- dlmo.DeviceCapability.DLERoles enumerates the DL role profiles supported by the DLE, where a value of TRUE indicates that the DLE supports the DL role profile. See 9.1.16 for a discussion of DLE roles.
  - Index 0 indicates whether the DLE supports the I/O role profile.
  - Index 1 indicates whether the DLE supports the router role profile.
  - Index 2 indicates whether the DLE supports the backbone router role profile.
  - Index 3 indicates whether the DLE supports the radio silence role profile. See 9.1.15.4.
  - Indices 4..7 are reserved and shall be set to FALSE.
- dlmo.DeviceCapability.EnergyDesign and EnergyLeft are described in 9.1.17 and 9.4.2.22.
- dlmo.DeviceCapability.DAckTurnaround indicates the time needed by the DLE to process a received data DPDU and respond with an ACK or NAK DPDU, in units of 2^{-15} s. All DLEs shall be capable of using the default timeslot templates (see Table 165, Table 166, and Table 167).

  The DAckTurnaround value is an upper bound on the externally-measured time interval required by the device to respond to a signal from its associated PHY that DPDU reception has completed and to initiate PHY transmission of any immediately following ACK/NAK DPDU. This measurement involves noting the time when the last symbol of a PhPDU corresponding to the initial DPDU of a transaction is presented to the receiving device and the first PhPDU signaling from that device is detected, where the transaction template used is a unicast receive template with the ACK/NAK DPDU timing referenced to the end of the just-received Data DPDU.
- dlmo.DeviceCapability.NeighborDiagCapacity indicates the capacity, in octets, of the NeighborDiag attribute. Only octets shown in Table 187 are included in NeighborDiagCapacity. The system manager indirectly creates OctetStrings in NeighborDiag by setting DiagLevel fields in the Neighbor attribute. The system manager should not configure a DLE to fill NeighborDiag in excess of its stated capacity, and a DLE may fail to accumulate data if the system manager exceeds this stated capacity. A value of 0x7FFF indicates that the capacity is sufficient to collect all available diagnostics for each dlmo.Neighbor.
- dlmo.DeviceCapability.RadioTransmitPower is the DLE's maximum supported power level, in dBm EIRP, that the DLE can legally support in the DLE's regulatory domain, as determined by dlmo.CountryCode. See 9.1.15.5, 9.1.15.6, 9.4.2.19 and Annex V.
- dlmo.DeviceCapability.SupportedCCAmodes is a list of CCA modes that the device supports (see 9.1.9.4.3):
  - Index 0 (Bit0) indicates whether CCA Mode 1 is supported.
  - Index 1 (Bit1) indicates whether CCA Mode 2 is supported.
  - Index 2 (Bit2) indicates whether CCA Mode 3 is supported.
  - Index 3 (Bit3) indicates whether CCA Mode 4 is supported.
  - Indices 4..7 (Bits4..7) are reserved and shall be set to FALSE.
- dlmo.DeviceCapability.ConstructionOptions indicates optional features that the device supports by construction:
  - Index 0 (Bit0) indicates whether group codes are supported in dlmo.Neighbor.

– Index 1 (Bit1) indicates whether graph extensions are supported in dlmo.Neighbor.

– Index 2 (Bit2) indicates whether the device is capable of receiving duocast or N-cast ACK/NAK DPDUs.

– Index 3 (Bit3) indicates whether the device is capable of supporting dlmo.Superframe.SfType=1, which may be needed in some regions for regulatory compliance.

– Index 4 (Bit4) indicates whether the device is capable of supporting the dlmo.Graph.Queue field which, when set to a non-zero value, reserves queue buffer capacity.

  Such queue capacity should not be so reserved unless dlmo.DeviceCapability.QueueCapacity exceeds the minimum requirement for a DLE's role profile (see Table B.8).

– Indices 5..7 (Bits5..7) are reserved and shall be set to FALSE.

9.4.2.24 Candidate neighbors

The dlmo.Candidates attribute is used to provide the system manager with a list of candidate neighbors. The DLE autonomously populates this attribute as it receives advertisements from a number of candidate neighbors. This attribute is then forwarded to the system manager so that routing decisions can be made. The system manager may reset dlmo.Candidates.N=0, thus signaling to the DLE to clear its history of received advertisements and resume the neighbor discovery process.

The attribute dlmo.DiscoveryAlert (Table 149) provides the system manager with control over neighbor discovery and reporting. The system manager sets dlmo.DiscoveryAlert, and later receives a copy of the dlmo.Candidates attribute through the dlmo.NeighborDiscovery alert. Alternatively, the system manager can read the dlmo.Candidates attribute on its own schedule, or arrange to report it periodically through the HRCO.

Table 149 – dlmo.DiscoveryAlert fields

Field name	Field encoding
Descriptor	Type: Alert report descriptor (see Table 269) Default: [FALSE, 0]
Duration	Type: Unsigned16 Units: 1 s Default: 60

Table 150 illustrates the structure of DiscoveryAlert.

Table 150 – dlmo.DiscoveryAlert structure

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Alert report disabled							
1	Alert report priority							
2	Duration							

When dlmo.DiscoveryAlert is enabled (Descriptor.Disabled=FALSE), the DLE shall report the contents of dlmo.Candidates.Duration seconds later using the dlmo.NeighborDiscovery alert. Once the DLE completes the alert, the DLE resets dlmo.DiscoveryAlert to zero.

When the NeighborDiscovery alert is triggered by the state of Duration (dlmo.DiscoveryAlert.Duration), the DLE shall automatically set Duration to zero, thereby resulting in a single NeighborDiscovery alert each time Duration is set to a non-zero value.

dlmo.DiscoveryAlert shall be enabled and default to Duration=60 when the DLE enters the joined state. This default indicates that the DLE shall, when it enters the joined state, accumulate information from advertisements in dlmo.Candidates for a period of 60 s, and then report the results using the dlmo.NeighborDiscovery alert. See 9.1.14.6. The system manager may override this default by writing to dlmo.DiscoveryAlert in conjunction with the join response.

Advertisements from neighboring routers include links that can be used to communicate with that router. When DiscoveryAlert is enabled, the DLE may use these links to interrogate, with a Data DPDU carrying a null DSDU, each candidate router, on multiple channels, and receive signal quality metric in the DAUX. This information enables the DEL to build a more accurate picture of signal quality in dlmo.Candidates.

When dlmo.DiscoveryAlert is disabled, the DLE should nonetheless passively build a dlmo.Candidates list as a background process after it joins the D-subnet, based on whatever advertisements it happens to receive in the course of operating the DLE's state machine.

If there is a dlmo.DL_Connectivity alert and DiscoveryAlert is enabled, the DLE shall also send a dlmo.NeighborDiscovery alert at the same time.

When dlmo.DiscoveryAlert.Duration is set to 0, the DLE shall not send dlmo.NeighborDiscovery alerts on a timed basis. The DLE should continue to maintain the Candidates attribute so that it can be read as needed by the system manager by reading the dlmo.Candidates attribute directly or by configuring the DLE to report it periodically through the HRCO.

The process of scanning for advertisements is described in 9.1.13.

dlmo.Candidates comprises a series of fields that are described in Table 151. In essence, the <Candidate>,<RSQI> tuple is repeated N times, providing an indication of signal quality to multiple neighbors.

dlmo.Candidates may reasonably exclude current entries in the dlmo.Neighbor table, when the same information for those neighbors is available through the neighbor diagnostics.

Table 151 – dlmo.Candidates OctetString fields

Field name	Field encoding
N (count of discovered neighbors)	Type: Unsigned8
$Neighbor_1$ (16-bit address of first candidate)	Type: ExtDLUint
$RSSI_1$ (radio signal strength of first candidate)	Type: Integer8
$RSQI_1$ (radio signal quality of first candidate)	Type: Unsigned8
…	…
$Neighbor_N$ (16-bit address of N^{th} candidate)	Type: ExtDLUint
$RSSI_N$ (radio signal strength of N^{th} candidate)	Type: Integer8
$RSQI_N$ (radio signal quality of N^{th} candidate)	Type: Unsigned8

Table 152 illustrates the structure of Candidates.

Table 152 – dlmo.Candidates structure

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	N							
1..2	$Neighbor_1$							
1	$RSSI_1$							
1	$RSQI_1$							
…	…							
1..2	$Neighbor_N$							
1	$RSSI_N$							
1	$RSQI_N$							

Fields include:

- dlmo.Candidates.N is the number of neighbors that have been discovered. A DLE shall support at least five (5) candidate entries. If many neighbors are discovered, the DLE may report only the best candidates based on the quality of the radio link.
- dlmo.Candidates.$Neighbor_N$ is the 16-bit address of each candidate neighbor in the D-subnet.
- dlmo.$RSSI_N$ indicates the strength of the radio signal in dBm from each candidate neighbor, based on received advertisements and possibly other DPDUs. See 9.1.15.2 for description of RSSI, including the fixed dBm bias within the reported measurement values.
- dlmo.$RSQI_N$ indicates the quality of the radio signal from each candidate neighbor, based on received advertisements and possibly other considerations. A higher number indicates a better radio signal. See 9.1.15.2. for description of RSQI.

9.4.2.25 Smoothing factors

The dlmo.SmoothFactors provides the smoothing factors for the dlmo.NeighborDiag attribute. The use of these factors is described in 9.1.15.3.

Three fields in dlmo.NeighborDiag involve exponential smoothing: RSSI, RSQI, and ClockBias. The smoothing factor α (alpha) for each of these values is individually configurable. Alpha is expressed as an integer percentage in the range of 0..100. RSSI and RSQI default to 10 %, so the values tend to reflect recent history. ClockBias defaults to 1 %, since that value is intended to show clock bias over an extended period of time.

Each smoothing factor involves three values: x_AlphaHigh, x_AlphaLow, and x_Threshold, for differing x. If the new data is below the threshold, use x_AlphaLow as the smoothing factor; otherwise use x_AlphaHigh.

The fields in dlmo.SmoothFactors are described in Table 153.

Table 153 – dlmo.SmoothFactors OctetString fields

Field name	Field encoding	Default
RSSI_Threshold (threshold for RSSI)	Type: Integer16	0
RSSI_AlphaLow (AlphaLow for RSSI)	Type: Unsigned8	10
RSSI_AlphaHigh (AlphaHigh for RSSI)	Type: Unsigned8	10
RSQI_Threshold (threshold for RSQI)	Type: Integer16	0
RSQI_AlphaLow (AlphaLow for RSQI)	Type: Unsigned8	10
RSQI_AlphaHigh (AlphaHigh for RSQI)	Type: Unsigned8	10
ClockBias_Threshold (threshold for ClockBias)	Type: Integer16	0
ClockBias_AlphaLow (AlphaLow for ClockBias)	Type: Unsigned8	1
ClockBias_AlphaHigh (AlphaHigh for ClockBias)	Type: Unsigned8	1

Table 154 illustrates the structure of SmoothFactors.

Table 154 – dlmo.SmoothFactors structure

Number of octets	Bits							
	7	6	5	4	3	2	1	0
2	RSSI_Threshold							
1	RSSI_AlphaLow							
1	RSSI_AlphaHigh							
2	RSQI_Threshold							
1	RSQI_AlphaLow							
1	RSQI_AlphaHigh							
2	ClockBias_Threshold							
1	ClockBias_AlphaLow							
1	ClockBias_AlphaHigh							

9.4.2.26 dlmo.QueuePriority

9.4.2.26.1 General

dlmo.QueuePriority is an attribute that enables the system manager to specify the nominal buffer capacity in the DLE's forwarding queue for specific priority levels. As described in 9.1.8.5, the system manager may configure a DLE's nominal buffer capacity to limit the number of buffers that can be used to forward low-priority Data DPDUs. For example, the system manager may configure dlmo.QueuePriority so that no more than 3 buffers shall be used to forward Data DPDUs with priority ≤ 2.

The nominal capacity of the forwarding queue can be found in dlmo.DeviceCapability.QueueCapacity (see 9.4.2.23).

9.4.2.26.2 Semantics

Table 155 specifies the fields for dlmo.QueuePriority.

Table 155 – dlmo.QueuePriority fields

Field name	Field encoding
N (count of priorities specified, 0..15, default N=0)	Type Unsigned8
$Priority_1$ (first priority)	Type: Unsigned8
$QMax_1$ (first buffer capacity)	Type: Unsigned8
…	
$Priority_N$ (N^{th} priority)	Type: Unsigned8
$QMax_N$ (N^{th} buffer capacity)	Type: Unsigned8

For example, if Priority is 2 and QMax is 3, then no more than 3 queue buffers shall be used to forward Data DPDUs with priority ≤ 2. This count shall not include Data DPDUs that are using queue capacity that was set aside for Data DPDUs being forwarded along a particular graph, based on dlmo.Graph[ ].Queue. Priority shall be enumerated in increasing order, so that $Priority_X$ shall be less than $Priority_{X + 1}$. Similarly, $QMax_X$ shall be less than $QMax_{X + 1}$.

The count $QMax_X$ sets the maximum available slots available for low-priority messages, not reserved slots for low-priority messages. In effect, $QMax_X$ ensures that the remainder of the queue is available for DPDUs of priority 1 + $Priority_X$.

As described in 9.2.2, the DE indicator in a Data DPDU header indicates whether a Data DPDU on the queue is eligible to be discarded in favor of an incoming DPDU of higher priority.

The default, where N = 0, indicates that the system manager has not configured a limit on the number of forwarding buffers that can be used for low priority DPDUs.

Table 156 illustrates the structure of dlmo.QueuePriority.

Table 156 – dlmo.QueuePriority structure

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	N							
2 (opt)	$Priority_1$							
3 (opt)	$QMax_1$							
…	…							
2N (opt)	$Priority_N$							
2N+1 (opt)	$QMax_N$							

9.4.2.27 dlmo.ChannelDiag

9.4.2.27.1 General

dlmo.ChannelDiag is a read-only dynamic attribute that reports DPDU transmit failure rates on each radio channel supported by this standard. This enables the system manager to be aware of consistent connectivity problems on a per-channel basis, as a diagnostic for spectrum management in a D-subnet.

Two diagnostics are reported for each channel, each indicating a different type of failure. $NoACK_N$ indicates the percentage of time that unicast DPDU transmissions on channel N did

not result in reception of an ACK/NAK DPDU. $CCABackoff_N$ indicates the percentage of time that the device aborted a transaction-initiating transmission on channel N due to CCA.

Under some situations, CCA backoff is part of normal D-subnet operation and is not indicative of poor channel quality. In particular, contention for shared links will lead to CCA backoff. Therefore, the DLE may be selective in its counting of CCA backoff. It is recommended that CCA backoff should not normally be counted when it occurs in shared links within slotted-channel-hopping superframes. Shared links are described in 9.1.8.2.

ChannelDiag values are 8-bit signed integers.

- A value of 0 indicates that no transmission has been attempted on the channel.
- Positive values in the range of 1..101 indicate the percentage failure rate plus one. For example, a $CCABackoff_6$ value of 26 indicates that 25 % of transaction-initiating transmissions on channel 6 were aborted due to CCA.
- Negative values in the range of -1 to -101 indicate the percentage failure rate as a negative number minus one. Failure rates are reported as negative numbers if they are based on 5 or fewer attempted transmissions. For example, a $NoACK_8$ value of -34 indicates that 33 % of unicast transmissions on a particular channel did not receive an acknowledgment, and that 5 or fewer transmissions have been attempted on that channel.

The DLE may selectively skip transmission links in anticipation of a failed transmission, as described in 9.1.7.2.4. Such skipped links should be treated as equivalent to NAK for the applicable channel.

Each time ChannelDiag is read by the system manager or reported periodically through the HRCO, its underlying accumulators shall be reset to zero.

9.4.2.27.2 Semantics

Table 157 specifies the fields for dlmo.ChannelDiag.

Table 157 – dlmo.ChannelDiag fields

Field name	Field encoding
Count (number of attempted unicast transmissions for all channels)	Type: Unsigned16
$NoACK_0$ (percentage of time transmissions on channel 0 did not receive an ACK DPDU)	Type: Integer8
$CCABackoff_0$ (percentage of time transmissions on channel 0 aborted due to CCA)	Type: Integer8
…	…
$NoACK_{15}$ (percentage of time transmissions on channel 15 did not receive an ACK DPDU)	Type: Integer8
$CCABackoff_{15}$ (percentage of time transmissions on channel 15 aborted due to CCA)	Type: Integer8

Table 158 illustrates the structure of dlmo.ChannelDiag.

Table 158 – dlmo.ChannelDiag structure

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Count							
1	$NoACK_0$							
1	$CCABackoff_0$							
…	…							
1	$NoACK_{15}$							
1	$CCABackoff_{15}$							

9.4.3 DLMO attributes (indexed OctetStrings)

9.4.3.1 General

Indexed OctetString management object attributes are structured collections of data, similar in concept to SQL tables. For example, the DLE maintains a list of neighbors in an indexed OctetString attribute called dlmo.Neighbor, where each neighbor can be visualized as a row in a table, with the structure of each row as shown in Table 168 and Table 169. A 15-bit index for each row, in this case corresponding to the 16-bit address of the neighbor, is provided for each indexed OctetString attribute in the DLMO.

For consistency of processing, all indexed OctetString attributes in the DL include an index in the first field that is type ExtDLUint. However, in the case of dlmo.Ch, dlmo.TsTemplate, and dlmo.Superframe, the index is limited to a range of 1..127, thus guaranteeing that the index can be represented in a single octet. References to these structures can also be compressed to a single octet.

Indexed OctetString index of zero shall not be used in the DLMO, except to indicate a null entry.

Figure 90 illustrates the relationship among some of the indexed OctetString DLMO attributes. Referential relationships are shown with arrows. For example, dlmo.Link refers to dlmo.TsTemplate, so an arrow is shown pointing in the direction of the reference. This roughly corresponds to a one-to-many relationship, where one template can be referenced by multiple links.

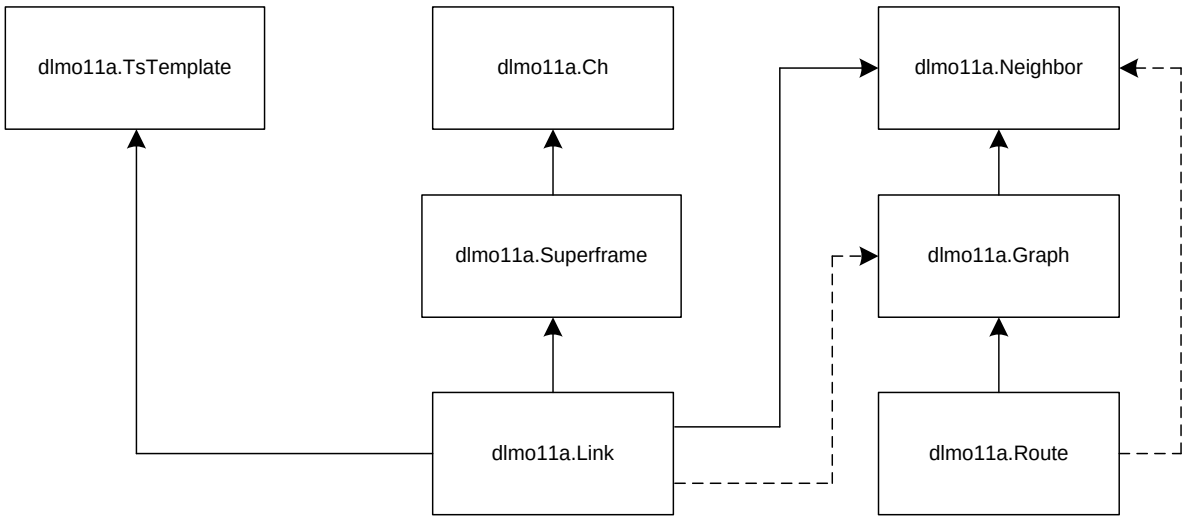


Figure 90 – Relationship among DLMO indexed attributes

As shown in Figure 90, dlmo.TsTemplate is referenced by dlmo.Link. Timeslot templates specify transaction structure and timing when a link is used.

dlmo.Link, dlmo.Superframe, and dlmo.Ch are related. Superframes select a channel-hopping pattern and a cyclic schedule. Links describe the various actions that are taken during each superframe cycle. Each link refers to one superframe.

For unicast (and duocast) transactions, dlmo.Link refers to dlmo.Neighbor. A single reference may encompass a group of neighbors. If a link is reserved for use of a certain graph, or gives preferential access to a certain graph, then dlmo.Link will also refer to dlmo.Graph. Since it is possible to configure a link without a reference to a graph, this reference is shown as a dotted line in Figure 90.

dlmo.Route and dlmo.Graph are related in that routes usually refer to graphs. dlmo.Graph and dlmo.Neighbor are related in that graphs refer to neighbors. dlmo.Route and dlmo.Neighbor are loosely related (indicated by a dotted line in Figure 90) in that source routing may implicitly refer to a neighbor.

An additional relationship, not shown in Figure 90, exists between dlmo.Neighbor and dlmo.NeighborDiag. Both are indexed by the 16-bit address of the DLE's neighbors, but with different purposes.

- dlmo.Neighbor: The system manager provides this static indexed OctetString attribute to enable the DLE to communicate with its immediate neighbors.
- dlmo.NeighborDiag: The system manager configures this dynamic indexed OctetString attribute to enable the DLE to collect and periodically report diagnostics for its immediate neighbors.

9.4.3.2 dlmo.Ch

9.4.3.2.1 General

dlmo.Ch is an indexed OctetString collection that contains available channel-hopping patterns.

Channel-hopping patterns 1 through 5 are reserved as standard defaults, as described in 9.1.7.2.5. Additional channel-hopping patterns may be added.

Each DLE can store multiple channel-hopping patterns, with a unique index for each pattern. Advertisements assume that channel-hopping patterns for the joining process are configured in the DLE when the advertisement is received. Thus any channel-hopping pattern referenced in an advertisement shall match one of the defaults.

The system manager inserts, updates, or deletes channel-hopping patterns by sending the DMAP a channel-hopping pattern, along with a unique index and (if selected) a TAI cutover time.

For a given channel-hopping pattern, the standard provides a mapping between the DL channel numbers and the more general IEEE 802.15.4 MAC channel numbers. As applied to the 2,4 GHz DSSS IEEE 802.15.4 radio, channel numbers shall be limited to the range of 0..15, corresponding to IEEE 802.15.4 channel numbers 11..26 (channel page 0), in the same order. Channel-hopping patterns for this radio shall not exceed a size of 16.

NOTE dlmo.Ch data types are limited to radios with 16 or fewer channels. Future radios with more channels would involve providing support for a less compressed representation.

Five default channel-hopping patterns are reserved and defined by this standard. Default channel-hopping patterns are enumerated and described in 9.1.7.2.5.

9.4.3.2.2 Semantics

Table 159 specifies the fields for dlmo.Ch. An index, used to identify each channel-hopping pattern, is consistent with DL conventions for indexed OctetStrings.

Table 159 – dlmo.Ch fields

Field name	Field encoding
* Index	Type: ExtDLUint (used as an index) Valid range: 1..127
Size	Type: Unsigned8 Valid range: 1..16
Seq (channel-hopping pattern, with Size entries)	Type: SEQUENCE OF Unsigned4 (SIZE (Size))

Table 160 illustrates the structure of dlmo.Ch.

Table 160 – dlmo.Ch structure

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Index (1..127)							
1	Size							
(Size+1)/2	Seq_1				Seq_0			
	Seq_3				Seq_2			
	...				...			
	0x0000 when Size is an odd number; Seq_{Size-1} when Size is an even number				$Seq_{2(n-2)}$			

9.4.3.3 dlmo.TsTemplate

9.4.3.3.1 General

dlmo.TsTemplate is an indexed OctetString collection that contains timeslot templates. Timeslot templates describe D-transaction timing.

Timeslot templates 1, 2, and 3 shall be reserved as standard defaults, as enumerated in Table 165, Table 166, and Table 167. Additional timeslot templates may be added by the system manager.

The system manager inserts, updates, or deletes timeslot templates by sending the DMAP a template, along with a unique index and (if selected) a TAI cutover time.

Template time offsets are specified in units of 2^{-20} s, which is approximately 1 μs. Timeslot duration is set by the superframes that use the template.

Template types include receive and transaction initiator templates. Both template types include acknowledgments for unicast transactions. The same templates can also be used for broadcast links, such as solicitations, that don't need acknowledgments and don't use the ACK/NAK DPDU timings.

Templates can be defined on three levels:

- Default templates are defined in the standard. These are the timeslot templates needed for joining (Table 165, Table 166, and Table 167). Template index=1 describes a generic receive transaction, and is used for receiving join responses. Template index=2 describes a generic transmit transaction, and is used for transmitting join requests. Template index=3 describes a transaction that operates its receiver for an entire timeslot, intended for operations such as scanning for advertisements or receiving loosely-timed slow-channel-hopping DPDUs. These generic templates shall be used during provisioning and joining, and may also be used for other purposes.
- Provisioned templates may be added during the provisioning process, with a lifetime that lasts until the DLE has joined the D-subnet. See 9.1.14.4.
- Subnet-specific templates may be provided to the DLE after the joining process is completed.

Data DPDU transmission timing is based on the transaction initiator's internal clock. ACK/NAK DPDU timing is specified as a time offset from a reference point that is indicated in the template. Usually, the configured time range for a transaction initiator is narrower than the time range for the transaction receivers, to account for guard times.

Timeslot templates shall always provide a reception window of at least 192 μs, which implies that DLEs shall be capable of controlling transmission timing with an accuracy of at least ±96 μs, i.e., ±6 PHY symbol periods.

By convention, timeslot template timing is specified based on the start and end times of DPDUs. PhPDU timing, dependent on the details of the physical layer that contains the DPDU, can be inferred from DPDU times. DPDU start time, as specified in timeslot templates, uses a convention that the DPDU begins at the instant just before the first octet in the DPDU header is transmitted. This convention applies to Data DPDU transmissions as well as ACK/NAK DPDUs.

NOTE In actual implementations based on IEEE 802.15.4 (2,4 GHz), PhL timing signals sometimes are triggered when an SFD (start frame delimiter) is completely transmitted or received. In such cases, the start time of the DPDU is 1 octet, or 32 μs, after that reference point.

ACK/NAK DPDU response time is commonly specified in relation to the end of the immediately preceding and triggering Data DPDU, with the ACK/NAK DPDU starting approximately 1 ms thereafter.

Alternatively, ACK/NAK DPDUs may be timed in relation to the scheduled end of the timeslot. By placing ACK/NAK DPDUs at a fixed offset within the timeslot, it is possible to meet regulatory requirements that would prohibit transmission of an ACK/NAK DPDU after a very short Data DPDU, where the sender of the ACK/NAK DPDU had also transmitted near the end of the prior timeslot.

Such timeslot-end-relative placement also supports routers that operate on multiple channels at the same time, sharing a common antenna, or devices whose antennas are in close proximity to each other. In both cases, without such scheduled alignment, Data DPDUs and/or ACK/NAK DPDUs transmitted on one channel might unintentionally jam time-overlapping reception whose reception is being attempted on another channel.

When ACK/NAK DPDU times are defined as offsets from the end of the timeslot, the time offsets, which are unsigned integers, shall be interpreted as referring to a time prior to the end of the timeslot.

Transaction receiver templates specify:

- The time range when the first octet of a Data DPDU can be received, indicating a time range to enable and disable the radio's receiver. A time range that exceeds the timeslot's duration indicates that the range extends only to the scheduled end of the timeslot.

- ACK/NAK DPDU delay time range.

A transaction responder for a unicast transaction should respond with an ACK/NAK DPDU as early as possible within this range, with the exception of intentionally staggered N-cast ACK/NAK DPDUs. ACK/NAK DPDU delay time may be specified in reference to the end of the received DPDU or as a backward offset from the scheduled end of the timeslot.

- Whether it is acceptable to operate past the timeslot boundary once reception of a DPDU begins, essentially extending timeslot duration.

Overrun of a slot boundary can be used to accommodate slow-channel-hopping, where transaction-originating DLEs may have a time-skewed sense of the slot boundaries. Support of such transactions originated by such DLEs involves allowing reception even when the transaction overruns a slot boundary.

Transaction initiator templates specify:

- Time range to begin transmission. A compliant DLE may begin its transmission at any time within the time range, based on its internal DLE clock. The time range is based on the start time of the Data DPDU. The CCA operation, if any, and the PhPDU's SHR and PHR precede that event and therefore may occur prior to the earliest permitted time to begin transmission.
- ACK/NAK DPDU delay time range. A transaction responder usually will respond as early as possible within this range, per its timeslot template but subject to regulatory constraints on the minimal required delay since the last prior transmission by the same device. ACK/NAK DPDU delay time may be specified in relation to the end of reception of the Data DPDU or the scheduled end of the timeslot.
- Indication of the CCA mode to be used before transmission to check for competing or ongoing transmissions.
- Indication of whether the DLE should periodically continue operating its receiver for the entire ACK/NAK DPDU delay time range, even after receiving an ACK/NAK DPDU (intended for duocast coverage testing; see 9.1.9.4.7).

9.4.3.3.2 Semantics

There are two variations of dlmo.TsTemplate, one for a transaction receiver template and another for a transaction initiator template. The variations are distinguished by a 2-bit type that is the first element. Type=0 indicates a transaction receiver template, and type=1 indicates a transaction initiator template. Types 2 and 3 are reserved for future use.

Table 161 specifies the fields for the transaction receiver template.

Table 161 – Transaction receiver template fields

Field name	Field encoding
* Index	Type: ExtDLUint (used as an index) Valid range: 1..127
Type (indicates that this is a transaction receiver template)	Type: Unsigned2: Named values: 0: transaction receiver template 1: see Table 163; other choices are reserved
AckRef (indicates reference for ACK/NAK DPDU time range)	Type: Unsigned1: Named values: 0: offset from end of incoming DPDU; 1: negative offset from end of timeslot
RespectBoundary (specifies whether or not slot boundaries shall be respected, i.e., whether a D-transaction may extend past a slot boundary)	Type: Boolean1: FALSE: slot boundaries do not need to be respected; TRUE: slot boundaries shall be respected
Reserved (octet alignment)	Type: Unsigned4=0
WakeupDPDU (earliest time when DPDU reception can be expected to begin, indicating when to enable radio's receiver; offset from timeslot's scheduled start time)	Type: Unsigned16 Units: 2^{-20} s
TimeoutDPDU (latest time when DPDU reception can be expected to begin, indicating when to disable receiver if no incoming DPDU is detected; offset from timeslot's scheduled start time)	Type: Unsigned16 Units: 2^{-20} s
AckEarliest (start of ACK/NAK DPDU delay time range, with start of that DPDU (PhSDU) being the time reference. Semantics depend on AckRef: if AckRef=1, subtract value from scheduled end time of timeslot to determine AckEarliest)	Type: Unsigned16 Units: 2^{-20} s
AckLatest (end of ACK/NAK DPDU delay time range, with start of that DPDU (PhSDU) being the time reference. Semantics depend on AckRef: if AckRef=1, subtract value from scheduled end time of timeslot to determine AckLatest)	Type: Unsigned16 Units: 2^{-20} s

Table 162 specifies the transaction receiver template structure.

Table 162 – Transaction receiver template structure

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	* Index (range 1..127)							
1	Type=0		AckRef	RespectBoundary	Reserved = 0			
2	WakeupDPDU							
2	TimeoutDPDU							
2	AckEarliest							
2	AckLatest							

Table 163 specifies the fields for the transaction initiator template.

Table 163 – Transaction initiator template fields

Field name	Field encoding
* Index	Type: ExtDLUint (used as an index) Valid range: 1..127
Type (indicates that this is a transaction initiator template)	Type: Unsigned2: Named values: 0: see Table 161; 1: transaction initiator template; remaining elements are reserved
AckRef (indicates reference for ACK/NAK DPDU delay time range)	Type: Unsigned1: Named values: 0: positive offset from end of transmitted/received DPDU 1: negative offset from end of timeslot
CCAmode (indicates whether to check CCA before transmission)	Type: unsigned2 (see 9.1.9.4.3): Named values: 0: CCA mode 4; 1: CCA mode 1; 2: CCA mode 2; 3: CCA mode 3
KeepListening (indicates whether the DLE should periodically continue operating its receiver until the end of the timeslot, even after reception of an ACK/NAK DPDU; see 9.1.9.4.7)	Type: Boolean1
Reserved (octet alignment)	Type: Unsigned2=0
XmitEarliest (earliest time to start DPDU transmission; offset from timeslot's scheduled start time)	Type: Unsigned16 Units: 2^{-20} s
XmitLatest (latest time to start DPDU transmission; offset from timeslot's scheduled start time)	Type: Unsigned16 Units: 2^{-20} s
WakeupAck (earliest time when reception of an ACK/NAK DPDU can be expected to begin; enable receiver early enough to receive an ACK/NAK DPDU beginning at this time. Semantics depend on ACKref; if AckRef=1, subtract value from scheduled end of timeslot to determine WakeupAck)	Type: Unsigned16 Units: 2^{-20} s
TimeoutAck (latest time when reception of an ACK/NAK DPDU can be expected to begin. DLE may disable receiver if ACK/NAK DPDU reception has not started by this time. Semantics depend on ACKref; if AckRef=1, subtract value from scheduled end of timeslot to determine TimeoutAck)	Type: Unsigned16 Units: 2^{-20} s
NOTE An AckEarliest value of 402 (384 μs) accommodates the IEEE 802.15.4 SIFS requirement of 6 octets, plus 6 octets for a PhPDU's SHR and PHR prior to the start of the ACK/NAK DPDU's PhSDU.	

Table 164 specifies the transaction initiator template structure.

Table 164 – Transaction initiator template structure

Number of octets	bits							
	7	6	5	4	3	2	1	0
1	* Index (range 1..127)							
1	Type		AckRef	CheckCCAmode		KeepListening	Reserved = 0	
2	XmitEarliest							
2	XmitLatest							
2	WakeupAck							
2	TimeoutAck							

Table 165, Table 166, and Table 167 specify the values for the three default DLE timeslot templates. These read-only timeslot templates use indexes 1, 2, and 3, and shall be used for links that are specified in advertisements. Their structure is general purpose, and they may be referenced by other links as well.

The DLE shall be capable of transmitting and receiving ACK/NAK DPDUs in the 1 032 μs ± 100 μs timing specified in these default templates, or more slowly if so specified in an alternative timeslot template. The attribute dlmo.DeviceCapability.DAckTurnaround informs the system manager if a device is capable of handling ACK/NAK DPDUs more quickly.

These default templates are required for the initial device provisioning and subnet joining processes. There is no requirement for any non-join use in an operational system.

Table 165 – Default transaction responder template, used during joining process

Field	Default value	Explanation
* Index	1	—
Type	0	—
AckRef	0	—
RespectBoundary	1	—
WakeupDPDU	1 271	1 212 μs
TimeoutDPDU	3 578	3 412 μs
AckEarliest	977	932 μs
AckLatest	1 187	1 132 μs

Table 166 – Default transaction initiator template, used during joining process

Field	Default value	Explanation
* Index	2	—
Type	1	—
AckRef	0	—
CCAmode	1	—
KeepListening	0	—
XmitEarliest	2 319	2 212 μs
XmitLatest	2 529	2 412 μs
WakeupAck	977	932 μs
TimeoutAck	1 187	1 132 μs

Table 167 – Default transaction responder template, used during joining process

Field	Default value	Explanation
* Index	3	—
Type	0	—
AckRef	0	—
RespectBoundary	0	If DPDU reception commences within the timeslot boundaries, complete processing of transaction
WakeupDPDU	0	Start of timeslot; allowing for timeslots that are contiguous. In the first timeslot of a contiguous series, a device may insert setup time at the start of the first timeslot not to exceed 1 271 μs
TimeoutDPDU	0xFFFF	End of timeslot
AckEarliest	977	Same as default transaction responder template
AckLatest	1 187	Same as default transaction responder template

9.4.3.3.3 Default template timings

Default timeslot templates are intended to match the timeslot structure of IEC 62591.

The default data DPDU transmission time is based on an offset of 2 312 μs from the start of the timeslot to the start of the data DPDU. The default transaction initiator template accounts for ± 100 μs of transaction initiator jitter, resulting in a default range of 2 312 μs ± 100 μs. The default transaction receiver template accounts for the same ± 100 μs of transmit jitter, plus clock drift of ± 1 000 μs, resulting in a receive range for the data DPDU of 2 312 μs ± 1 100 μs.

The default ACK/NAK DPDU transmission time is based on an offset of 1 032 μs from the end of the data DPDU to the start of the ACK/NAK's DPDU. The default slot transaction templates allow for ± 100 μs of transmit jitter, for a default template of 1 032 μs ± 100 μs. Clock drift is considered inconsequential in the short time span from the end of a data DPDU to the start of an immediately-following ACK/NAK DPDU.

In the IEEE 802.15.4 radio used in this standard, the physical layer header is 6 octets, or 192 μs, in duration. Thus, radio transmission or reception begins 192 μs earlier than the times specified in the templates. CCA, if required, precedes the radio startup.

Templates do not account for the internals of radio operation, leaving that as an internal device matter. For example, the value of 932 μs for WakeupACK means that the physical layer header is allowed to begin transmission 192 μs sooner, or as early as 932 μs -192 μs = 740 μs following the end of the DPDU. The receiver of the ACK/NAK DPDU, also with a nominal WakeupAck of 932 μs, therefore needs its radio to be running at 740 μs following the end of the DPDU and ready to start receiving the ACK/NAK DPDU at that time. If the receiver needs additional time to account for radio startup and receiver jitter, then the radio needs to be enabled sufficiently in advance of 740 μs to ensure that it can receive the full physical layer header.

9.4.3.3.4 Considerations for required minimum inter-transmission gap

Some regulatory regimes require that there be a minimum time period after one transmission ceases before the device is again permitted to transmit. It is the responsibility of each DLE to note the time that the most recent transmission ceased, to use that information to determine the earlierst moment that the device may again transmit, and to refrain from activating its transmitter before that moment.

NOTE This can be achieved by recording separately the ending time of the most recent transmission, adding a claimed-operating-mode-dependent Tx-gap-time constant, then comparing that resultant time with the projected

time of the next transmission to determine whether the transmission is permitted under the applicable regulations. See Annex V.

While the system manager can configure the various dlmo.TsTemplates to provide behavior in accordance with these regulatory requirements, it is the responsibility of the individual device to ensure that the requirements are observed no matter the configuration. dlmo.CountryCode (9.1.15.6, 9.4.2.19) provides the configuration information necessary to determine which regulatory aspects are in force.

9.4.3.4 dlmo.Neighbor

9.4.3.4.1 General

dlmo.Neighbor is an indexed OctetString collection that contains information about immediate unicast neighbors. dlmo.Neighbor entries are referenced by graphs and links.

The system manager inserts, updates, or deletes dlmo.Neighbor entries by sending the DMAP neighbor entries, along with a unique index and (if selected) a TAI cutover time. The neighbor's 16-bit address is used as an index.

The neighbors in the dlmo.Neighbor attribute are set by the system manager, not by the DLE itself. The DLE autonomously builds a list of candidate neighbors in the dlmo.Candidates attribute, as described in 9.4.2.24. This list is then forwarded to the system manager. The system manager considers the radio connectivity that is reported in dlmo.Candidates, but may also consider other criteria such as resource constraints, historical performance, or D-subnet topology.

When the DLE processes an advertisement during the join procedure, the DLE automatically adds the advertising router as an entry in the dlmo.Neighbor table, thereby enabling communication through the proxy. This entry persists after the DLE successfully joins the D-subnet, unless it is deleted or updated by the system manager. The system manager infers the existence and content of this entry based on the identity of the proxy that the DLE uses to join the D-subnet. See 9.3.5.2.1.

This standard follows the IETF RFC 4944 convention, whereby 16-bit unicast addresses are limited to the range of 0x 0xxx xxxx xxxx xxxx. See 9.3.3.6.

Diagnostic information related to neighbors can be found in the attribute dlmo.NeighborDiag (see 9.4.3.9).

9.4.3.4.2 Semantics

Table 168 specifies the fields for dlmo.Neighbor.

Table 168 – dlmo.Neighbor fields

Field name	Field encoding
* Index (DL16Address of the neighbor)	Type: ExtDLUint (used as an index) Valid range: 1..32 767
EUI64 (EUI64Address of the neighbor)	Type: EUI64Address
GroupCode1 (associate a group code with a set of neighbors; used by dlmo.Link)	Type: Unsigned8
GroupCode2 (associate a group code with a set of neighbors; used by dlmo.Link)	Type: Unsigned8
ClockSource (indicates whether neighbor is a DL clock source)	Type: Unsigned2: Named values: 0: not a clock source 1: secondary 2: preferred 3: reserved
ExtGrCnt (count of graphs virtually extended for this neighbor)	Type: Unsigned2
DiagLevel (selection of neighbor diagnostics to collect)	Type: BooleanArray2 Named indices: 0: collect link diagnostics 1: collect clock diagnostics
LinkBacklog (indicates that link information is provided to facilitate clearing message queue backlog to the neighbor)	Type: Unsigned1; Named values: 0: no extra link information 1: extra link information is provided
Reserved (octet alignment)	Type: Unsigned1=0
ExtendGraph	Type: SEQUENCE OF Octet2 (SIZE ExtGrCnt) – octet pairs Valid range: See Table 170
LinkBacklogIndex (activate this link to clear queue backlog)	Type: ExtDLUint Valid range: 1..127 Null and not transmitted if LinkBacklog=0 Link index if LinkBacklog=1
LinkBacklogDur (duration of link activation)	Type: Unsigned8 Units: Link occurrences Null and not transmitted if LinkBacklog=0 1..255 if LinkBacklog=1
LinkBacklogActivate (link activation criterion)	Type: Unsigned8 Units: DPDUs on queue Null and not transmitted if LinkBacklog=0 1..255 if LinkBacklog=1

Table 169 illustrates the structure of dlmo.Neighbor.

Table 169 – dlmo.Neighbor structure

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1–2	* Index							
8	EUI64							
1	GroupCode1							
1	GroupCode2							
1	ClockSource		ExtGrCnt		DiagLevel		LinkBacklog	Reserved
2 × ExtGrCnt	$ExtendGraph_1$							
	...							
	$ExtendGraph_{ExtGrCnt}$							
0..1	LinkBacklogIndex							
0..1	LinkBacklogDur							
0..1	LinkBacklogActivate							

Fields include:

- EUI64. In order to communicate with a neighbor, the DSC needs the EUI64Address of that neighbor. This information is stored in the Neighbor table. It is populated by the system manager, with one exception. During the joining and provisioning processes, where the neighbor entry is created automatically from the advertisement, the EUI64Address shall be acquired by the DLE through the ACK/NAK DPDU as described in 9.1.10.1.
- GroupCode1, GroupCode2. Links with a matching group code may be used to address this neighbor. The scope of the group code is within a single DLE. When a link has a group NeighborType=2, the link designates a group code instead of a neighbor, and the link applies to any queued DPDU where the neighbor has a matching group code. This enables a single transmit link to be shared by a group of neighbors. A value of zero indicates that no group code applies. Support for group codes is mandatory in routers but is a construction option in I/O devices. The presence or absence of this capability is reported to the system manager when the DLE joins the D-subnet through the field dlmo.DeviceCapability.ConstructionOptions (see 9.4.2.23).
- ClockSource. If this indicator is > 0, then the neighbor shall be a DL clock source for this DLE. A value of 1 indicates a secondary DL clock source, and a value of 2 indicates a preferred DL clock source. See 9.1.9.2.3.
- ExtGrCnt and ExtendGraph. See 9.1.6.3 for a discussion of graph extensions. See Table 170 for the fields in each graph extension entry in ExtendGraph. If the neighbor's address matches the destination address encoded in the DADDR subheader, and the Graph_ID designated in ExtendGraph matches the DPDU's DL route, extend the designated graph to that neighbor for that DPDU. For each neighbor, up to three such graphs may be extended. Support for ExtendGraph is a device construction option, and the presence or absence of this capability is reported to the system manager when the DLE joins the D-subnet through the field dlmo.DeviceCapability.ConstructionOptions (see 9.4.2.23).

For each ExtendGraph entry, include:

– Graph ID is the 12-bit graph ID that is being extended. See 9.4.3.6.

– LastHop. If this indicator is 1, the DLE shall only use links to the neighbor for applicable DPDUs. In this case, the DLE shall treat the extended graph index as the single forwarding alternative.

– PreferredBranch. If this indicator is 1, the DLE should treat this graph extension as the preferred branch for applicable DPDUs. (See PreferredBranch field in 9.4.3.6.2).

Table 170 specifies the fields for each element in the ExtendGraph sequence.

Table 170 – ExtendGraph fields

Field name	Field encoding
Graph_ID (*Index of dlmo.Graph attribute)	Type: Unsigned12
LastHop (indicates whether the neighbor shall be the last hop)	Type: Boolean1
PreferredBranch (indicates whether to treat the neighbor as the preferred branch)	Type: Boolean1
Reserved (octet alignment)	Type: Unsigned2=0

Graphs are extended implicitly whether ExtendGraph is designated or not. The optional explicit ExtendGraph feature is intended to support optimizations that seek to control the graph ID of this final hop, and/or designate it as the last hop or preferred branch.

– DiagLevel. If this indicator has any non-zero bits, then the DLE shall collect link diagnostics for this neighbor in the read-only attribute dlmo.NeighborDiag. If Bit0=1, summary diagnostics shall be accumulated. If Bit1=1, clock diagnostics shall be accumulated. See 9.4.3.9.

– LinkBacklog, LinkBacklogIndex, LinkBacklogDur, and LinkBacklogActivate provide an index to a link that may be activated through the DAUX subheader (type 2). See 9.3.5.4 for a general description of link activation. The DLE, when transmitting a DPDU to this neighbor, may detect a backlog of applicable DPDUs on its message queue, and therefore signal the neighbor to activate the link Index=LinkBacklogIndex to receive DPDUs for the next LinkBacklogDur occurrences of the link (see 9.3.5.4). LinkBacklogActivate indicates the size of the applicable DPDU backlog on the queue, excluding the DPDU being transmitted, that should trigger sending the link activation DAUX, unless the link is already activated.

The system manager, when it configures LinkBacklogIndex, LinkBacklogDur, and LinkBacklogActivate, should also configure a transmit link with the same index, so that a receive link on the neighbor has the same index as a corresponding transmit link in the DLE that originates the link activation message, with both being activated for the number of occurrences indicated by LinkBacklogDur. DPDUs queued to this neighbor should be given high priority for that link index during the period defined by LinkBacklogDur. When counting candidate DPDUs on the message queue, the DLE should account for all queued DPDUs that can be addressed to the neighbor.

The transaction initiator (the DLE that initiates activation of the idle links) should activate its own idle link for a count of LinkBacklogDur transmission opportunities (link occurrences). The receiver (the DLE on which the receive idle link has been activated) should activate its idle link for LinkBacklogDur reception opportunities (link occurrences). Transmission and reception opportunities should be counted even if a higher priority link is actually used in a particular timeslot. Generally, the idle link should be configured with a high priority.

Table 171 specifies the ExtGraph structure.

Table 171 – ExtGraph structure

Number of octets	bits							
	7	6	5	4	3	2	1	0
1	Graph_ID (bits 11..4)							
1	Graph_ID (bits 3..0)				LastHop	PreferredBranch	Reserved = 0	

9.4.3.4.3 dlmo.NeighborDiagReset

dlmo.NeighborDiagReset provides read/write access to the field of the dlmo.Neighbor attribute that is used to set the neighbor diagnostic level. It is conceptually similar to an SQL view of dlmo.Neighbor. It can be used to read and write a specific field within dlmo.Neighbor, but it shall not be used to add or delete entries.

Table 172 specifies the fields within a dlmo.NeighborDiagReset OctetString.

Table 172 – dlmo.NeighborDiagReset fields

Field name	Field encoding
* Index	Type: ExtDLUint (used as an index) Valid range: 1..32 767
Reserved (octet alignment)	Unsigned4 = 0
DiagLevel (selection of neighbor diagnostics to collect)	Type: BooleanArray3; Named indices: 0: collect summary information 1: collect clock information
Reserved (octet alignment)	Type: Unsigned2 = 0

Table 173 illustrates the structure of dlmo.NeighborDiagReset.

Table 173 – dlmo.NeighborDiagReset structure

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1..2	* Index							
1	Reserved = 0				DiagLevel		Reserved = 0	

Fields are exactly as specified for the identically named fields in the dlmo.Neighbor attribute, and the instantaneous values of these fields are the same as in dlmo.Neighbor.

9.4.3.5 dlmo.Superframe

9.4.3.5.1 General

dlmo.Superframe is an indexed OctetString collection that contains superframes. The superframe structure enables the system manager to connect a set of links (dlmo.Link) to a repeating schedule of timeslots (dlmo.Superframe) of fixed duration. The superframe also designates a baseline channel-hopping schedule for those links.

The system manager inserts, updates, or deletes dlmo.Superframe entries by sending the DMAP a superframe, along with a unique index and, if selected, a TAI cutover time.

Each superframe describes a schedule that is specified in reference to TAI time zero. Derivation of current timeslot state from the superframe definition is described in 9.4.3.5.3.

A superframe may be configured with randomized timings, intended exclusively for links that transmit or receive solicitations and/or advertisements. Since a randomized superframe schedule is not synchronized to its neighbors, such a superframe shall not be used to transmit payload in DSDUs.

9.4.3.5.2 Semantics

Table 174 specifies the fields for dlmo.Superframe.

Table 174 – dlmo.Superframe fields

Field name	Field encoding
* Index	Type: ExtDLUint (used as an index) Valid range: 1..127
TsDur (duration of timeslots within the superframe; timeslots are realigned with TAI time reference every 250 ms)	Type: Unsigned16 Units: 2^{-20} s
ChIndex (selects channel-hopping pattern from dlmo.Ch)	Type: ExtDLUint
ChBirth (absolute slot number where channel channel-hopping pattern nominally started)	Type: Unsigned8
SfType (type of superframe)	Type: Unsigned2; Named values: 0: baseline 1: hop on link only 2: randomize slow-hop duration 3: randomize superframe period
Priority (priority to select among multiple available links)	Type: Unsigned4
ChMapOv (indicates whether to override ChMap default)	Type: Boolean1; Valid range: FALSE: ChMapOv not transmitted, so defaults to 0x7FFF; TRUE: ChMapOv transmitted and used
IdleUsed	Type: Boolean1; Valid range: FALSE: IdleTimer not transmitted, so defaults to -1; TRUE: IdleTimer transmitted and used
SfPeriod (base number of timeslots in each superframe cycle)	Type: Unsigned16 Valid range: > 0
SfBirth (absolute slot number where the first superframe cycle nominally started)	Type: Unsigned16
ChRate (indicates the number of timeslots per hop)	Type: ExtDLUint Valid range: 0 = invalid 1 = slotted-channel-hopping > 1 = slow-channel-hopping [a]
ChMap (channel map used to eliminate certain channels from the channel-hopping pattern, to limit the frequency spectrum in use)	Type: Unsigned16 or null
IdleTimer (idle/wakeup timer for superframe)	Type: Integer32 or null See text
RndSlots (indicates extent of randomization, in number of slots)	Type: Unsigned8 or null
a In some regulatory regimes, as specified via dlmo.CountryCode (9.4.2.19), the maximum value of ChRate is constrained to be no greater than (400 ms / TsDur).	

Table 175 illustrates the structure of dlmo.Superframe.

Table 175 – dlmo.Superframe structure

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	* Index							
2	TsDur							
1	ChIndex (range 1..127)							
1	ChBirth							
1	SfType		Priority				ChMapOv	IdleUsed
2	SfPeriod							
2	SfBirth							
1..2	ChRate							
0 or 2	ChMap							
0 or 4	IdleTimer							
0 or 1	RndSlots							

Fields include:

- Timeslot duration (dlmo.Superframe[ ].TsDur). All timeslots within a superframe have the same duration, and a DLE is not required to handle multiple timeslot durations at the same time. Timeslots shall be realigned to the TAI clock every 250 ms. See 9.1.9.1 for information on timeslot alignment.
- Channel-hopping pattern identifier (dlmo.Superframe[ ].ChIndex). Select an available pattern from dlmo.Ch (see 9.4.3.2.2).
- Channel-hopping birthday (dlmo.Superframe[ ].ChBirth). Specifies the starting point of the channel-hopping pattern, as a timeslot offset from TAI=0. Calculation of current position in hop sequence is described in 9.4.3.5.3.
- Superframe type (dlmo.Superframe[ ].SfType) indicates the type of superframe. Handling of each superframe type is described in 9.4.3.5.3.
- Superframe priority (dlmo.Superframe[ ].Priority) indicates the priority of the superframe. A higher Priority value will give that superframe link a higher priority. See 9.1.8.5 (pseudocode) for additional information on priority levels.
- Channel map override default (dlmo.Superframe[ ].ChMapOv). Indicates whether to override ChMap default of 0x7FFF. If ChMapOv=1, change the default based on ChMap.
- Superframe period (dlmo.Superframe[ ].SfPeriod) indicates the number of timeslots in each base cycle of the superframe. With 10 ms timeslots, a 16-bit superframe period supports superframes up to about 10 min long.
- Superframe birthday (dlmo.Superframe[ ].SfBirth) indicates the nominal starting point for the “first” superframe that began its first cycle soon after TAI=0. It provides the slot offset from absolute timeslot 0, which occurred at nominal time TAI=0. SfBirth shall equal ChBirth when SfType= 1. See 9.4.3.5.3.
- Channel-hopping rate (dlmo.Superframe[ ].ChRate) indicates the number of timeslots per hop. A channel-hopping rate of 1 indicates slotted-channel-hopping; a channel-hopping rate greater than 1 indicates some degree of slow-channel-hopping. ChRate shall =1 when SfType=1.
- Channel map (dlmo.Superframe[ ].ChMap) is used to eliminate certain channels from the channel-hopping pattern, thus shortening the channel-hopping pattern in use. Bit positions 0..15 correspond to channels 0..15, where a 0 bit in any position indicates that the corresponding channel shall not be used by the superframe, with the channel-

hopping sequence shortened accordingly. This attribute shall not be transmitted and defaults to 0x7FFF if ChMapOv=0 (i.e., optional channel 15 is excluded by default).

- Idle superframe timer (dlmo.Superframe[ ].IdleUsed, dlmo.Superframe[ ].IdleTimer) provides the system manager with control over when a superframe is activated or idle, where an idle superframe treats all of its links as idle. The system manager may set IdleTimer through the dlmo.Superframe attribute or the dlmo.SuperframeIdle attribute. IdleTimer is not transmitted and defaults to a value of -1 if IdleUsed=0.
- When IdleTimer is set to a positive number, the superframe shall be active and IdleTimer shall be decremented by the DLE each TAI second until it reaches a value of 0. When the value of IdleTimer is 0, the superframe shall be idle.
- When IdleTimer is set to a negative number that is less than -1, the superframe shall be idle and IdleTimer shall be incremented by the DLE each TAI second until it reaches a value of -1. When the value of IdleTimer is -1, the superframe shall be active.

Randomization of superframes is controlled by dlmo.Superframe[ ].RndSlots. RndSlots is meaningless and shall not be transmitted if dlmo.Superframe[ ].SfType<2. Randomized superframes are intended exclusively to enable randomized D-subnet discovery processes. For example, a DLE in the provisioned state may be configured with a randomized superframe that is used to search for advertisements from a target D-subnet. Such randomization can be used to guarantee that the scan's sleep cycle is not synchronized with the advertisement schedule of the target D-subnet, thereby ensuring that an advertisement is eventually received. Only receive links, dedicated advertisements, and solicitations should be configured for use with a superframe where RndSlots>0. All other links for such randomized superframes shall be treated as idle.

– When SfType=2, a randomized number of timeslots in the range of 0 to RndSlots shall be added to the end of each superframe cycle.

– When SfType=3, a randomized number of timeslots in the range of 0 to RndSlots shall be added to the duration of each slow-channel-hopping period. If slotted-channel-hopping is used, each hop shall be treated as a slow-channel-hopping period extended by a randomized number of timeslots.

9.4.3.5.3 Superframe current timeslot state

The current superframe timeslot state shall be derived from the superframe fields as described herein. This description is not intended to constrain implementations, but only results. All features described herein shall be supported by all DLEs that comply with this standard, unless specifically designated as a construction option.

These derivations of the current timeslot state use fields in dlmo.Superframe[ ], which are based on the state at TAI time of zero or soon thereafter. Implementations may reasonably use these formulas to establish a starting state when the superframe is initialized, and then update that state incrementally going forward. However, the incremental update approach will not work when there is a change in any field used in the base calculation, or in fields in other attributes (dlmo.Ch[ ] in general, and dlmo.Link[ ] for SfType=1) that are used in the base calculation. When those fields are changed, the current state needs to be derived again.

NOTE The notion of an absolute timeslot is used here as a variable to calculate the current timeslot state. Other standards use an absolute timeslot to identify the timeslot. This standard uses an absolute timeslot only as an intermediate value in a calculation; it is not referenced elsewhere in this standard.

Each TAI quarter-second period has a fixed number of timeslots that can be described by the formula:

$$\text{SlotsPer0\_25s} = \text{floor}((2^{18}\ \text{s}) / \text{TsDur})$$

where floor(x) is the largest integer not greater than x.

An absolute slot number (SlotNumAbs) can be derived from the scheduled start time of the current timeslot (ScheduledTaiTime), simplified by the fact that ScheduledTaiTime is required to be re-aligned to TAI time every quarter-second. The slot offset from TAI=0 can be derived accordingly:

Tai0_25sStart = floor(ScheduledTaiTime / (0,25 s)) × (0,25 s)

SlotWithin0_25s = (ScheduledTaiTime – Tai0_25sStart) / TsDur

SlotNumAbs = ((Tai0_25sStart / (0,25 s)) × SlotsPer0_25s) + SlotWithin0_25s

SfType=0 designates the baseline case, where all superframe cycles include a fixed number of timeslots and the channel-hopping schedule also has a fixed cycle.

The superframe provides a fixed superframe period (SfPeriod) which is the number of timeslots in each superframe cycle. It also provides an absolute slot number (SfBirth), following TAI=0, as a reference starting time for the first superframe. The superframe offset of the current timeslot is:

SfOffset = (SlotNumAbs – SfBirth) mod SfPeriod

where (x mod y) equals (x – (floor(x / y) × y)) for positive y.

The channel-hopping pattern nominally begins at absolute slot number ChBirth. The number of elements in the channel-hopping schedule (ChCount) can be determined from the size of the channel-hopping pattern selected by ChIndex, and by subtracting the number of entries that are removed from the sequence as indicated by ChMap.

The number of timeslots in a cycle of the channel-hopping pattern depends on whether slow-channel-hopping or slotted-channel-hopping is used, as indicated by ChRate.

ChCycle = ChCount × ChRate

The timeslot offset into that channel-hopping cycle is:

ChOffset = (SlotNumAbs – ChBirth) mod ChCycle

SfType=1 designates a variant of slotted-channel-hopping, where channel-hopping occurs only when there is a link.

Device support for SfType=1 is a construction option, as reported to the system manager through the attribute dlmo.DeviceCapability.ConstructionOptions (bit 5). SfType=1 shall not be combined with slow-channel-hopping, i.e., ChRate=1.

The superframe offset is as described in SfType=0.

The number of channel hops per superframe cycle (ChPerSuperframe) is determined by counting the number of timeslots in each superframe cycle that are referenced by at least one link. This is not a simple count of links, because some links may refer to multiple timeslots, and some timeslots may be referenced by multiple links.

Since the number of channel hops is a multiple of the number of superframes, the next step is to calculate the number of superframe cycles that have been completed since TAI=0.

CurrentSfStartAbs = (SlotNumAbs – SfOffset)

SfCyclesSinceBirth = (CurrentSfStartAbs – SfBirth) / SfPeriod

For SfType=1, the superframe cycle and channel-hopping cycles are required to start at the same time (SfBirth=ChBirth). The channel offset at the start of the current superframe is a function of the number of mapped channels (see 9.4.2.12). It is determined by the formula:

$ChOffset_{StartSf}$ = (SfCyclesSinceBirth × ChPerSuperframe) mod NumMappedChannels

where NumMappedChannels = $\sum \text{dlmo.Superframe[ ].ChMap}_k$ for the specific superframe.

Starting from $ChOffset_{StartSf}$, the current channel-hopping-offset can be determined by stepping through the superframe from the start of the superframe cycle to the current timeslot. The channel-hopping-offset within each superframe cycle cannot be reduced to a simple linear formula since the links are not necessarily spread evenly through the cycle.

SfType=2 extends each slow-channel-hopping interval by a randomized number of timeslots in the range of 0 to dlmo.Superframe[ ].RndSlots. The initial starting point of the channel-hopping sequence may also be randomized when SfType=2, and superframe timing shall be as described for SfType=0.

SfType=3 extends each superframe cycle by a randomized number of timeslots in the range of 0 to dlmo.Superframe[ ].RndSlots. The initial starting point of the superframe cycle should also be randomized when SfType=3, and the channel-hopping sequence shall be as described for SfType=0.

9.4.3.5.4 Slow-channel-hopping

Slow-channel-hopping is defined as a superframe where dlmo.Superframe[ ].ChRate>1, resulting in a set of contiguous links on the same channel. The channel-hopping rate, ChRate, may be configured as equal to the superframe period, SfPeriod, and in that case each channel-hopping period may reasonably be configured as a range of links using dlmo.Link[ ].Schedule=2.

The receive side of a slow-channel-hopping configuration should use the default transaction receiver template for scanning as per Table 167, or a similar template. This template, when applied to contiguous timeslots on the same channel, should run its receiver continuously, and may run a transaction to completion even if that transaction runs across the edge of a timeslot. A receive link using that template may repeat frequently or continuously within a superframe, usually with a low priority to give precedence to slotted-channel-hopping operations.

A set of slow-channel-hopping receive links on a given channel, using the default transaction receiver template for scanning, may be temporarily interrupted by higher-priority transactions, for example, as shown graphically in Figure 66. In the absence of such transactions, the receiver should run continuously across the timeslot boundaries in such configurations.

The transmit side of a slow-channel-hopping configuration should use a template appropriate for a transmit transaction on the D-subnet, such as the default transaction initiator template as per Table 166. The transmit link configuration should be configured to account for clock drift. For example, in a D-subnet with 10 ms timeslots, a particular device in a slow-channel-hopping configuration might be expected to experience clock drift of up to 15 ms between clock updates. In that example, the first and last two timeslots in each slow-channel-hopping period should not be designated as transmit links, thereby incorporating 20 ms guard times into the configuration.

The transmit side in a slow-channel-hopping configuration may designate specific timeslots for transmission, or alternatively it may designate a range of timeslots. When a range of timeslots is designated, the channel-hopping rate should match the superframe period, and the transmit link should be designated as a range (dlmo.Link[ ].Schedule=2), shown graphically in Figure 72. In that configuration, the DLE should treat each range as a single transmit opportunity, and select the transmit link within the range on a randomized basis.

9.4.3.5.5 dlmo.SuperframeIdle

dlmo.SuperframeIdle provides read/write access to only the fields of the dlmo.Superframe attribute that relate to the idle superframe. It is conceptually similar to an SQL view of

dlmo.Superframe. It can be used to read and write specific fields within superframe indexed OctetStrings, but cannot be used to add or delete entries.

Table 176 specifies the fields within a dlmo.SuperframeIdle OctetString.

Table 176 – dlmo.SuperframeIdle fields

Field name	Field encoding
* Index	Type: ExtDLUint (used as an index)
Reserved (octet alignment)	Type: Unsigned7=0
IdleUsed (indicates whether the superframe is idle when the IdleTimer is zero)	Type: Boolean1
IdleTimer (idle/wakeup timer for superframe)	Type: Integer32 or null

Table 177 illustrates the structure of dlmo.SuperframeIdle.

Table 177 – dlmo.SuperframeIdle structure

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	* Index							
1	Reserved = 0							IdleUsed
0 or 4	IdleTimer							

Fields are exactly as specified for identically named fields in the dlmo.Superframe attribute, and the instantaneous values of these fields are identical to those in dlmo.Superframe.

9.4.3.6 dlmo.Graph

9.4.3.6.1 General

dlmo.Graph is an indexed OctetString collection that contains graphs. On a particular DLE, a graph is simply a list of neighbors that can be used for the next hop when a graph is specified in the DROUT subheader.

The system manager inserts, updates, or deletes dlmo.Graph entries by sending the DMAP a graph, along with a unique index and (if selected) a TAI cutover time.

Graph ID=0 shall not be used, because this value is reserved as an indicator in the DROUT subheader, as described in 9.3.3.6.

Graph IDs are limited to a range of 12-bit values, with a range of $1..2^{12}$. In source routing, these 12-bit graph IDs are encoded as 0x 1010 gggg gggg gggg.

As described in 9.1.6.3, immediate neighbors are implicitly treated as covered by a graph, whether the neighbor is listed in the graph structure or not. When the DPDU's destination address is in a DLE's neighbor table, the graph is automatically extended to cover that neighbor. Thus, even though the structure of dlmo.Graph can support only a few neighbors, the graph can handle many more through such graph extensions.

It is advisable for Graph IDs to be unique within the scope of a D-subnet. However, duplicated graph IDs are not prohibited. When two graphs with the same graph ID intersect in a single DLE, they unite to become a single graph.

9.4.3.6.2 Semantics

Table 178 specifies the fields for dlmo.Graph.

Table 178 – dlmo.Graph

Field name	Field encoding
* Index	Type: ExtDLUint (used as an index) Valid range: 1..4095
PreferredBranch (indicates whether to treat the first listed neighbor as the preferred branch)	Type: Boolean1
NeighborCount	Type: Unsigned3 Valid range: 0..4
Queue (allocates buffers in the message queue for DPDUs that are being forwarded using this graph)	Type: Unsigned4
MaxLifetime	Type: ExtDLUint
Neighbors (index into dlmo.Neighbor; usually two or three neighbors in list for next-hop diversity)	Type: SEQUENCE OF ExtDLUint (SIZE(NeighborCount))

Table 179 illustrates the structure of dlmo.Graph in the case where each ExtDLUint requires one octet.

Table 179 – dlmo.Graph structure

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1..2	* Index							
1	PreferredBranch	NeighborCount			Queue			
1..2	MaxLifetime							
1..2	$Neighbors_0$							
1..2	$Neighbors_1$							
...	...							
1..2	NeighborsNeighborCount-1							

Elements include:

- dlmo.Graph[ ].PreferredBranch. If this indicator is 1, treat the first listed neighbor as the preferred branch, and the DLE should wait until there is an opportunity to try at least one transmission along the preferred branch before attempting other alternatives. If this indicator is 0, do not give such preferential treatment to the first listed neighbor.
- dlmo.Queue allows the system manager to reserve up to 15 buffers of the message queue for DPDUs that are following the graph.
- dlmo.Graph[ ].MaxLifetime (units ¼ s). If this element is non-zero, the value of dlmo.MaxLifetime shall be overridden for all DPDUs being forwarded following this graph.
- List of neighbors (commonly two neighbors for next-hop link diversity).

9.4.3.7 dlmo.Link

9.4.3.7.1 General

dlmo.Link is an indexed OctetString collection that contains links. Each link refers to exactly one dlmo.Superframe entry.

The system manager inserts, updates, or deletes links by sending the DMAP a link, along with a unique index and (if selected) a TAI cutover time.

When a neighbor is referenced in a transmit link, DPDUs that refer to that neighbor are considered as candidates for the link. DPDUs that refer to the neighbor through the first entry in the DROUT subheader, either directly by address or indirectly through a graph, shall be considered as candidates for the link. In addition, DPDUs that designate the neighbor as the destination address in the DADDR subheader shall also be considered as candidates for the link. The exception is that certain links are designated exclusively for DPDUs following specific graphs, and only DPDUs with matching GraphIDs shall be considered as candidates for such links. If multiple DPDUs on the message queue are candidates for a given link, the DPDU is selected by priority as described in 9.1.8.5.

9.4.3.7.2 Semantics

Table 180 specifies the fields for dlmo.Link.

Table 180 – dlmo.Link fields

Field name	Field encoding
* Index	Type: ExtDLUint (used as an index)
SuperframeIndex (reference to dlmo.Superframe entry)	Type: ExtDLUint
Type (see Table 182 and associated text)	Type: Unsigned8
Template1 (dlmo.TsTemplate reference to primary template)	Type: ExtDLUint
Template2 (dlmo.TsTemplate reference to secondary template)	Type: ExtDLUint or null
NeighborType	Type: Unsigned2 Valid range: 0..2
GraphType	Type: Unsigned2 Valid range: 0..2
SchedType	Type: Unsigned2
ChType	Type: Unsigned1
PriorityType	Type: Unsigned1
Neighbor (identify neighbor or group)	Type: ExtDLUint or null
GraphID (12-bit identity of graph with exclusive or prioritized access to link)	Type: ExtDLUint or null
Schedule (link schedule; see Table 184)	Type: See Table 184
ChOffset (select channel based on offset from superframes hop pattern)	Type: Unsigned8 or null
Priority (link priority)	Type: Unsigned8 or null Valid range: 0x 0000 xxxx

Table 181 illustrates the structure of dlmo.Link. ExtDLUint fields are shown as single octets.

Table 181 – dlmo.Link structure

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1..2	* Index							
1	SuperframeIndex (range 1..127)							
1	Type							
1	Template1 (range 1..127)							
0 or 1	Template2 (range 1..127)							
1	NeighborType		GraphType		SchedType		ChType	PriorityType
0..2	Neighbor							
0..2	GraphID							
1..4	Schedule (see Table 184)							
0 or 1	ChOffset							
0 or 1	Priority							

Elements include:

- dlmo.SuperframeIndex. Indicates the superframe reference for the link.
- dlmo.Link[ ].Type. Indicates how the link is configured for transmission and/or reception, and/or neighbor discovery. See Table 182.
- dlmo.Link[ ].Template1. Primary timeslot template. See 9.4.3.3 for a discussion of templates.
- dlmo.Link[ ].Template2. Secondary timeslot template, for transmit/receive (T/R) slots only, in combination with other link selections. Use Template2 as the transaction receiver template, if there is no DPDU in the queue for the primary template. Template 2 is transmitted and meaningful only for TRx links, that is, links where Link[ ].Type bits 6 and 7 both have a value of 1. See 9.1.8.5.
- dlmo.Link[ ].NeighborType, and dlmo.Link[ ].Neighbor. A neighbor is designated for transmit links, and may be designated for duocast/N-cast receive links. See 9.4.3.4 for a discussion of neighbors. When a neighbor is designated in a link, it may reference either a dlmo.Neighbor index or a group (dlmo.Neighbor[ ].GroupCode).
  - If dlmo.Link[ ].NeighborType=0, dlmo.Link[ ].Neighbor is null, and not transmitted.
  - If dlmo.Link[ ].NeighborType=1, dlmo.Link[ ].Neighbor designates an index into the dlmo.Neighbor attribute.
  - If dlmo.Link[ ].NeighborType=2, dlmo.Link[ ].Neighbor designates a group.
- dlmo.Link[ ].GraphType, dlmo.Link[ ].GraphID. DPDUs following a particular graph may be given exclusive or priority access to certain transmit links. These fields, when so configured, limit link access to certain graphs, thereby connecting the link to a particular communication flow through the D-subnet. When GraphType is left blank, the transmit link is available to any DPDU that is being routed through the link's designated neighbor. When GraphType is used, a particular graph is given exclusive or priority access to the link.
  - If GraphType=0, the GraphID element is null and is not transmitted.
  - If GraphType=1, the link is designated for exclusive use by a particular graph. Access to the link shall be limited to DPDUs following that graph.
  - If GraphType=2, DPDUs with a matching graph ID are given priority access. DPDUs following that graph should be given priority over other DPDUs when the link is used.

- dlmo.Link[ ].SchedType, dlmo.Link[ ].Schedule. Indicates the timeslot position(s) of the link within each superframe cycle. The schedule may designate a fixed offset, a repeating set with a fixed interval, a range, or a bitmap, as follows:

  0: offset only;

  1: offset and interval;

  2: range;

  3: bitmap.

- dlmo.Link[ ].ChType, dlmo.Link[ ].ChOffset. Indicates how the link's channel is selected.
  - If dlmo.Link[ ].ChType=0, dlmo.Link[ ].ChOffset is null and not transmitted. Simply follow the superframe's baseline channel-hopping pattern.
  - If dlmo.Link[ ].ChType=1, add dlmo.Link[ ].ChOffset to the superframe's current dlmo.Superframe[ ].ChOffset, modulo the effective channel-hopping pattern size (after accounting for excluded channels) and select the channel accordingly.
- dlmo.Link[ ].PriorityType, dlmo.Link[ ].Priority. Indicates how the link's priority is set. Link priorities are functionally described in 9.1.8.5.
  - If dlmo.Link[ ].PriorityType=FALSE, dlmo.Link[ ].Priority is null and not transmitted. For transmit links, use priority dlmo.LinkPriorityXmit. For receive links, use priority dlmo.LinkPriorityRcv.
  - If dlmo.Link[ ].PriorityType=TRUE, use the dlmo.Link[ ].Priority. If the link is both a transmit link and a receive link, use dlmo.Link[ ].Priority for transmissions, and dlmo.LinkPriorityRcv for reception.

Table 182 illustrates the structure of the field dlmo.Link[ ].Type.

Table 182 – dlmo.Link[ ].Type structure

Octet	Bits							
	7	6	5	4	3	2	1	0
1	Transmit	Receive	Exponential backoff	Idle	Discovery: Named values: 0: none 1: advertisement 2: reserved 3: solicitation		JoinResponse: link used to send join response to neighbor	SelectiveAllowed

The link types are defined as follows:

- Bit 7: Transmit (T=TRUE). Indicates transmission of payload.
- Bit 6: Receive (R=TRUE).
- Bit 5: Exponential backoff (B=TRUE). Indicates whether the transaction originator should apply exponential backoff rules for retries (shared), versus using the timeslot without regard to exponential backoff (not shared). See 9.1.8.2.
- Bit 4: Idle (I=TRUE). When TRUE, the link shall be idle unless temporarily activated in conjunction with transmission of an activate DAUX subheader; see 9.3.5.4. When FALSE, link activation does not apply to the link. A timeslot designated as idle may include a neighbor reference. Even without a neighbor reference, it needs transmit and receive Booleans set as needed to refer to the timeslot template it will need when activated.
- Bits 3..2: Discovery (DD). Advertisement or solicitation configuration. DD='01' specifies a single advertisement transmitted with timing as defined in the timeslot template. If DD='01', the link should be used to transmit an advertisement or solicitation, even if there is no higher-order payload that needs to be sent. DD='11' distinguishes a solicitation from an advertisement. DD='10' shall be ignored.

NOTE DD='10' was used by ISA 100.11a for an alternate form of advertisement, and is thus not available for future assignment. Its use is not supported by this standard.

- Bit 1: JoinResponse (J=TRUE). When a router proxies a join request for an immediate neighbor, it will eventually receive a message from the system manager to forward to the DLE that is joining. The router forwards these messages using links that are flagged with JoinResponse=TRUE. A join response on the message queue can be identified by a DPDU header with a destination EUI64Address. A timeslot designated as supporting a join response may include a neighbor reference, thereby enabling the link to also be used for regular D-subnet traffic. Even without a neighbor reference, it needs the Transmit boolean to be set.
- Bit 0: SelectiveAllowed (S=TRUE). The DLE may, without system manager direction, autonomously and selectively treat transmit links as idle if they occur on certain radio channels with a history of poor connectivity. This is a form of selective channel utilization, and is described in 9.1.7.2.4. The DLE may skip links occurring on channels that it autonomously deems problematic due to a history of poor connectivity, potentially with the granularity of a specific channel used for communication with a specific neighbor. In this manner, the DLE can save energy and reduce unnecessary interference with other users of the spectrum. However, the DLE shall not skip links in this fashion when the link is flagged with SelectiveAllowed=0.

Table 183 shows allowed combinations of bits in the representation of dlmo.Link[ ].Type. Bits shown as X indicate that a value of FALSE (0) or TRUE (1) is allowed. For example, several combinations involving Transmit=TRUE show an X for Exponential backoff, indicating that such links might be configured as shared or not.

Table 183 – Allowed dlmo.Link[ ].Type combinations

Combination TRBI DDJS	Description
1X10 001X	Join response
10XX 000X	Transmit, no advertisement
11XX 000X	Transmit/receive, no advertisement
10X0 010X	Transmit, advertisement
0000 0100	Dedicated advertisement
0000 1100	Solicitation
010X 0000	Receive

A transmit/receive link combination is essentially a compressed representation of a transmit link and a separate receive link. If at least one outbound DPDU on the queue matches the link, it shall be treated as a transmit link using the primary timeslot template. Otherwise, it shall be treated as a receive link using the secondary timeslot template, with priority dlmo.LinkPriorityRcv.

It is possible to configure channel-hopping patterns, superframe periods and link intervals so that the result is that only certain radio channels in the channel-hopping sequence are actually used. For example, if a link repeats every 20^{th} timeslot, and the channel-hopping pattern includes 15 channels, then only three channels will actually be used by the link. To avoid such scenarios, the link interval and the channel-hopping pattern may be configured to have a greatest common divisor equal to one (1).

Table 184 specifies the different types of schedules for a given link. Links in a superframe are indexed based on the timeslot offset in each cycle, with the first timeslot in each cycle having an offset of zero.

Table 184 – Values for dlmo.Link[ ].Schedule

Value for dlmo.Link[ ].Schedule	Element encoding	Description
0 = offset only	ExtDLUint (offset)	Link occurs once at a fixed timeslot position in each superframe cycle
1 = offset and interval	ExtDLUint, ExtDLUint (offset, interval)	Link occurs multiple times in each cycle, first at the given offset and then repeating at an interval until the end of the cycle. Values are specified in number of timeslots
2 = range	ExtDLUint, ExtDLUint (first, last)	Link occurs at a range of slots in each superframe cycle, starting with the offset given by the first value and continuing until the offset given by the second value
3 = bitmap	BooleanArray32	Bitmap covers the first 32 timeslots in each superframe cycle. Link occurs in timeslots with a corresponding TRUE value in the array. Following LSB conventions, the array is transmitted in DMAP messages with indices 7..0 transmitted first and indices 31..24 transmitted last

9.4.3.8 dlmo.Route

9.4.3.8.1 General

dlmo.Route is an indexed OctetString collection that contains routes. dlmo.Route describes available routes for DPDUs. When a DSDU comes down the protocol stack from the NL, it receives a final destination address, along with a contract ID and a priority class. The DLE maps the contract ID and destination address into a route, based on table lookups. The priority class from the NL is simply copied to the DPDU header without being considered in route selection.

The system manager inserts, updates, or deletes routes by sending the DMAP a route, along with a unique index and (if selected) a TAI cutover time.

For a description of route selection, see 9.1.6.5.

9.4.3.8.2 Semantics

Table 185 specifies the fields for dlmo.Route.

Table 185 – dlmo.Route fields

Field name	Field encoding
* Index	Type: ExtDLUint (used as an index)
Size (size (number of entries) of Route attribute)	Type: Unsigned4 Valid range: 1..15
Alternative	Type: Unsigned2
Reserved (octet alignment)	Type: Unsigned2=0
ForwardLimit (initialization value for the forwarding limit in DPDUs that use this route)	Type: Unsigned8
Route (series of routing destinations; if entry high-order bit is 0, specifies a unicast address; if entry high-order bits are 0x1010, specifies a graph)	Type: SEQUENCE OF Unsigned16 (SIZE (size))
Selector (see text)	Type: Unsigned16 or null
SrcAddr (see text)	Type: ExtDLUint or null

Table 186 illustrates the structure of dlmo.Route.

Table 186 – dlmo.Route structure

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1..2	* Index							
1	Size				Alternative		Reserved = 0	
1	ForwardLimit							
2	$Route_0$							
...	...							
2	$Route_{Size-1}$							
0 or 2	Selector							
0..2	SrcAddr							

The attribute dlmo.Route[ ].Selector depends on the setting of dlmo.Route[ ].Alternative:

- When dlmo.Route[ ].Alternative=0, select this route if dlmo.Route[ ].Selector matches ContractID and dlmo.Route[ ].SrcAddr matches the SrcAddr (source address) field in the DADDR subheader. This alternative shall not be used unless the DLE is a backbone router. If dlmo.Route[ ].Alternative<>0, dlmo.Route[ ].SrcAddr is null and shall not be transmitted.
- When dlmo.Route[ ].Alternative=1, select this route if dlmo.Route[ ].Selector matches ContractID.
- When dlmo.Route[ ].Alternative=2, select this route if dlmo.Route[ ].Selector matches destination address.
- When dlmo.Route[ ].Alternative=3, use this route as the default. dlmo.Route[ ].Selector is null and shall not be transmitted.

dlmo.Route[ ].Alternative shall be applied in order, with lower numbered alternatives given precedence over higher numbered alternatives. There should be no more than one dlmo.Route entry per ContractID/SrcAddr combination (Alternative=0), no more than one entry per ContractID (Alternative=1), no more than one entry per destination address (Alternative=2), and no more than one default (Alternative=3). If there are duplicates, the matching entry with the lowest index shall be selected.

9.4.3.9 dlmo.NeighborDiag

9.4.3.9.1 General

dlmo.NeighborDiag is an indexed OctetString collection that contains diagnostics for a set of neighbors. The attribute is read-only, with rows created as needed by the DLE.

Each NeighborDiag entry comprises an array of one or two OctetStrings, with each entry corresponding to a different neighbor.

NeighborDiag entries are instantiated by the system manager, by setting dlmo.Neighbor[ ].DiagLevel bits to non-zero values. If and only if Bit0=1, then summary diagnostics shall be collected for the neighbor, consolidated across all channels. If and only if Bit1=1, then detailed clock diagnostics shall be collected for the neighbor, consolidated across all radio channels.

NOTE Individual channel diagnostics are collected through the attribute dlmo.ChannelDiag.

Diagnostics include counters and levels, that are accumulated as described in 9.1.15.3. Generally, counters are incremented by one in the course of successful or unsuccessful

transactions, while RSSI (signal strength) and RSQI (signal quality) are levels that are accumulated as exponential moving averages.

NeighborDiag is reported in three general ways:

- Through the HRCO, the system manager can configure the DLE to report NeighborDiag periodically, such as every 30 min. Following each such report, on a per-entry basis, NeighborDiag counts shall be reset to zero. Levels shall use the current value as a starting point for the next period.
- The system manager can read (poll) NeighborDiag as a read-only attribute on a per-entry basis. As in an HRCO report, counts shall be reset to zero when read.
- The DLE can additionally be configured, through the dlmo.AlertPolicy attribute, to report NeighborDiag information when diagnostic values exceed a threshold. Only the row triggering the alert is reported. No values are reset.

Generally in this standard, an indexed OctetString's metadata capacity is reported as the number of rows. Since rows in NeighborDiag can have substantially variable sizes, metadata for NeighborDiag (DiagMeta) shall be reported in memory capacity in octets for the OctetStrings, with the convention that each ExtDLUint field is assumed to consume two octets. A DLE shall have the capacity for summary diagnostics (dlmo.NeighborDiag[ ].Summary) for at least half of its neighbor capacity as indicated by dlmo.NeighborMeta.Capacity, or for at least two neighbors, whichever is greater.

9.4.3.9.2 Semantics

Each NeighborDiag entry includes three OctetStrings, one each for Summary and ClockDetail diagnostics. A zero-length OctetString indicates that the diagnostic is not being accumulated. Table 187 specifies the fields for dlmo.NeighborDiag.

Table 187 – dlmo.NeighborDiag fields

Field name	Field encoding
* Index	Type: ExtDLUint (neighbor address, used as an index)
Summary	Type: OctetString
ClockDetail	Type: OctetString

Table 188 specifies the fields within the diagnostic summary OctetString.

Table 188 – Diagnostic summary OctetString fields

Field name	Field encoding
RSSI (level)	Type: Integer8
RSQI (level)	Type: Unsigned8
RxDPDU (count)	Type: ExtDLUint
TxSuccessful (count)	Type: ExtDLUint
TxFailed (count)	Type: ExtDLUint
TxCCA_Backoff (count)	Type: ExtDLUint
TxNAK (count)	Type: ExtDLUint
ClockSigma (level)	Type: Integer16

Table 189 specifies the structure of the diagnostic summary OctetStrings.

Table 189 – Diagnostic summary OctetString structure

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	RSSI							
1	RSQI							
1 or 2	RxDPDU							
1 or 2	TxSuccessful							
1 or 2	TxFailed							
1 or 2	TxCCA_Backoff							
1 or 2	TxNAK							
2	ClockSigma							

Fields include:

- RSSI (signal strength): See 9.1.15.2 for discussion of RSSI units. RSSI is accumulated as an exponential moving average; see 9.1.15.3.
- RSQI (signal quality): See 9.3.5.5 and 9.1.15.2 for discussion of RSQI units. RSQI is accumulated as an exponential moving average; see 9.1.15.3.
- RxDPDU: Count of valid Data DPDUs received from neighbor, excluding DPDUs with null DSDU payloads (as well as AKC/NAK DPDUs).
- TxSuccessful: Count of successful unicast transmissions to the neighbor, where an ACK DPDU was received in response.
- TxFailed: Count of DPDU unicast transmissions, where an ACK/NAK DPDU was expected but not received in response.
- TxCCA_Backoff: Count of unicast transmissions that were aborted due to CCA. These aborted transmissions are not included in TxFailed.
- TxNAK: Count of NAK DPDUs received, not included in TxFailed.
- ClockSigma: A rough estimate, to within one standard deviation, of recent clock corrections, in units of 2^{-20} s. A one-sigma value accounts for approximately 68 % of clock corrections. ClockSigma is reset to zero whenever counters are reset.

NOTE 1 See 9.1.15.3 for the behavior of counters.

If the DLE autonomously treats a transmit link as idle, as described in 9.1.7.2.4, such skipped links shall not be counted in the neighbor diagnostics. However, such skipped links are reflected in channel diagnostics, as described in 9.4.2.27.

Table 190 specifies the fields within the diagnostic Clock OctetString.

Table 190 – Diagnostic ClockDetail OctetString fields

Field name	Field encoding
ClockBias (level, signed)	Type: Integer16
ClockCount (count)	Type: ExtDLUint
ClockTimeout (count)	Type: ExtDLUint
ClockOutliers (count)	Type: ExtDLUint

Table 191 specifies the structure of the diagnostic ClockDetail OctetString, with ExtDLUint fields shown as a single octet.

Table 191 – Diagnostic ClockDetail OctetString structure

Octets	Bits							
	7	6	5	4	3	2	1	0
2	ClockBias							
1 or 2	ClockCount							
1 or 2	ClockTimeout							
1 or 2	ClockOutliers							

If the neighbor is a preferred DL clock source, as indicated by IncludeClock=1, it is recommended that the DLE be configured to accumulate the ClockDetail fields.

ClockSigma, ClockBias, and ClockOutliers all relate to clock corrections. If the neighbor is a DL clock source, these values relate to clock corrections received from the DL clock source. If the neighbor is not DL clock source, these values relate to clock corrections sent to the DLE's neighbor through the ACK/NAK DPDU.

Fields include:

- ClockBias: An exponential moving average (EMA) of clock correction, in units of 2^{-20} s, including sign, with a 1 % smoothing factor. See 9.1.15.3 for a discussion of EMA.

NOTE 2 If this value is significantly non-zero it indicates that the clock is biased relative to the remote clock source.

- ClockCount: Count of clock updates received from or transmitted to the neighbor.
- ClockTimeout: Count of clock timeout events.
- ClockOutliers: Estimated count of clock corrections in excess of three standard deviations as per ClockSigma.

9.5 DLE methods

9.5.1 Method for synchronized cutover of DLE attributes

A Scheduled_Write method, with MethodID=1, is provided to set an attribute at a specific TAI time. It exactly follows the template found in Table J.1.

9.5.2 Methods to access indexed OctetString attributes

Various methods in the DLE relate to writing, reading, and deleting indexed OctetString attributes. These methods are generally based on the templates provided in Annex J.

All indexed OctetString attributes in the DL are indexed by a single integer encoded as an ExtDLUint (see 9.3.2.2). Following the convention of the template methods, these indexes are duplicated in the input arguments. For example, the Write_Row method includes an index as an input argument even though the index is also carried within the OctetString that constitutes the new entry.

Table 192 specifies the Read_Row method.

Table 192 – Read_Row method

Method name	Method ID	Method description			
Read_Row	2	Method to read the value of a single row of an indexed OctetString attribute whose data is visualized as an information table			
	Input arguments				
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**	
	1	Attribute_ID	Unsigned16	The attribute ID in the DLMO to which this method is being applied	
	2	Index	Unsigned16	The * Index field in the attribute to access a particular row	
	Output arguments				
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**	
	1	Data_Value	OctetString	An octet string that contains the contents of the row. If the row is empty, the OctetString shall contain only the Index, encoded as ExtDLUint	

Table 193 specifies the Write_Row method.

Table 193 – Write_Row method

Method name	Method ID	Method description			
Write_Row	3	Method to set/modify the value of a single row of an indexed OctetString attribute whose data is visualized as an information table			
	Input arguments				
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**	
	1	Attribute_ID	Unsigned16	The attribute ID in the DLMO to which this method is being applied, as determined by the ordinal index of the attribute in the DLMO definition	
	2	Scheduled_TAI_Time	Unsigned32	TAI time in seconds at which the value should be written to the row of the structured attribute. If the time is in the past, relative to the receiving device's time sense, the write shall be performed immediately	
	3	Index	Unsigned16	The * Index field in the attribute to access a particular row	
	4	Data_Value	OctetString	An octet string that contains the new contents of the row. If the DLMO row is unpopulated, a new row is created containing the OctetString if memory is available. If the DLMO row already exists, its contents are replaced with the OctetString. If the OctetString is null then the row shall be deleted	
	Output arguments				
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**	
	None				

Table 194 specifies the Write_Row_Now method. It is identical to the Write_Row method, without the Scheduled_TAI_Time argument. It has the effect of writing an indexed OctetString row immediately on receipt.

Table 194 – Write_Row_Now method

Method name	Method ID	Method description			
Write_Row_Now	4	Method to set/modify the value of a single row of an indexed OctetString attribute whose data is visualized as an information table			
	Input arguments				
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**	
	1	Attribute_ID	Unsigned16	The attribute ID in the DLMO to which this method is being applied, as determined by the ordinal index of the attribute in the DLMO definition	
	2	Index	Unsigned16	The * Index field in the attribute to access a particular row	
	3	Data_Value	OctetString	An octet string that contains the new contents of the row. If the DLMO row is unpopulated, a new row is created containing the OctetString if memory is available. If the DLMO row already exists, its contents are replaced with the OctetString. If the OctetString is null then the row shall be deleted	
	Output arguments				
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**	
	None				

9.6 DL alerts

9.6.1 DL_Connectivity alert

DLE performance diagnostics are accumulated in the attributes dlmo.NeighborDiag for per-neighbor diagnostics, and dlmo.ChannelDiag for per-channel diagnostics. Normally the system manager configures the HRCO to report these diagnostics periodically, and the DLE automatically resets the diagnostic counters whenever these attributes are so reported. Between such reports, diagnostics may indicate a problem that needs to be reported to the system manager immediately. The DL_Connectivity alert provides the mechanism for the DLE to report such issues, and the dlmo.AlertPolicy attribute enables the system manager to set thresholds for such reporting.

The attribute dlmo.AlertPolicy enables/disables the DL_Connectivity alert and provides thresholds to control whether alerts are reported. dlmo.AlertPolicy is an OctetString containing fields as shown in Table 195.

Table 195 – dlmo.AlertPolicy fields

Field name	Field encoding
Descriptor (enables or disables the DL_Connectivity alert)	Type Alert report descriptor Default: Disabled=TRUE Default: Priority=0
NeiMinUnicast (minimum number of unicast transactions needed for a neighbor report)	Type: ExtDLUint
NeiErrThresh (report neighbor diagnostic if the percentage error rate reaches this threshold)	Type: Unsigned8
ChanMinUnicast (minimum number of unicast transactions on a channel needed as a pre-condition for triggering an alert)	Type: ExtDLUint
NoAckThresh (report ChannelDiag if a NoAck value is greater than this threshold)	Type: Unsigned8
CCABackoffThresh (report ChannelDiag if a CCABackoff value is greater than this threshold)	Type: Unsigned8

Table 196 specifies the structure of the dlmo.AlertPolicy OctetString.

Table 196 – dlmo.AlertPolicy OctetString structure

Number of octets	Bits							
	7	6	5	4	3	2	1	0
2	Descriptor							
1 or 2	NeiMinUnicast							
1	NeiErrThresh							
1 or 2	ChanMinUnicast							
1	NoAckThresh							
1	CCABackoffThresh							

Fields include:

- dlmo.AlertPolicy.Descriptor determines whether or not the DL_Connectivity alert is enabled. By default, DL_Connectivity alert is disabled until the system manager enables it by populating this attribute with appropriate thresholds. When Disabled=TRUE, all other dlmo.AlertPolicy fields are meaningless and ignored.
- dlmo.AlertPolicy.NeiMinUnicast sets a minimum number of attempted unicast transactions before an error rate is considered significant. The count of attempted unicast transactions for a neighbor is the sum of the dlmo.NeighborDiag fields

  TxSuccessful+TxFailed+TxCCA_Backoff+TxNAK.

  If this sum is less than NeighborTxMinReport, do not send a DL_Connectivity alert for the neighbor.
- dlmo.AlertPolicy.NeiErrThresh sets the threshold for reporting a DL_Connectivity alert for a neighbor. The percentage error rate is calculated as:

  (TxFailed + TxCCA_Backoff + TxNAK) x 100 /
  (TxSuccessful + TxFailed + TxCCA_Backoff + TxNAK)

  If this value is greater than NeiErrThresh, the diagnostics for the neighbor should be reported using the DL_Connectivity alert unless there is an insufficient number of unicast transactions to the neighbor or the same alert has been recently reported.
- dlmo.AlertPolicy.ChanMinUnicast is similar to NeiMinUnicast. Counters underlying dlmo.ChannelDiag are not exposed, but a count of attempted unicast transactions is implicit in the reported ratios.

- dlmo.AlertPolicy.NoAckThresh and dlmo.AlertPolicy.CCABackoffThresh provide thresholds for reporting. Since the values reported by dlmo.ChannelDiag are ratios, the reported values are simply compared to the thresholds. If the value exceeds the thresholds, dlmo.ChannelDiag should be reported through the DL_Connectivity alert unless the ChanMinUnicast requirement is not met or the same alert has been recently reported.

The system manager may respond to the DL connectivity alert by collecting diagnostics to more fully characterize the situation. Alternatively, particularly if a modified topology is easily achieved, the system manager may simply reconfigure the D-subnet topology.

Table 197 illustrates the structure of the DL_Connectivity alert.

Table 197 – DL_Connectivity alert

Standard object type name: DL management object (DLMO)					
Standard object type identifier: 124					
Description of the alert: Poor neighbor connectivity					
Alert class (Enumerated: alarm or event)	**Alert category (Enumerated: device diagnostic, comm. diagnostic, security, or process)**	**Alert type (Enumerated: based on alert category)**	**Alert priority**	**Value data type**	**Description of value included with alert**
Event	Comm	0 = DL_Connectivity	Medium	Type: DL16Address	See Table 187

The format of the OctetString transmitted with the DL_Connectivity alert is shown in Table 198. It is simply the attribute number for either dlmo.ChannelDiag (48) or dlmo.NeighborDiag (46), followed by an OctetString containing the diagnostic data from that attribute. In the case of ChannelDiag, the entire attribute is transmitted. In the case of NeighborDiag, only the row that triggered the alert is transmitted, with the neighbor address specified within the row identifying the neighbor.

Table 198 – DL_Connectivity alert OctetString

Octets	**Bits**							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	AttributeNumber (Unsigned8)							
N	Attribute (OctetString)							

9.6.2 NeighborDiscovery alert

As described in 9.4.2.24, the NeighborDiscovery alert provides a mechanism for the DLE to report the contents of the OctetString in dlmo.Candidates attribute.

Table 199 illustrates the structure of the NeighborDiscovery alert.

Table 199 – NeighborDiscovery alert

Standard object type name: DL management object (DLMO)					
Standard object type identifier: 124					
Description of the alert: Neighbor discovery alert					
Alert class (Enumerated: alarm or event)	**Alert category (Enumerated: device diagnostic, comm. diagnostic, security, or process)**	**Alert type (Enumerated: based on alert category)**	**Alert priority**	**Value data type**	**Description of value included with alert**
Event	Comm	1 = NeighborDiscovery	Medium	Type: OctetString	An exact copy of the OctetString in dlmo.Candidates; see 9.4.2.24

10 Network layer

10.1 General

Clause 10 provides an overview of NL functionality. It also describes conceptual services that the NL offers to the layer above it (transport), the NL management object (NLMO), and the structure of NPDUs.

NOTE NPDU header formats have been designed for compatibility with IETF RFC 6282.

The NL follows the big endian convention; multi-octet fields are documented and transmitted with the high-order octet first (since they are treated as a series of octets by the lower layer). Within an octet, bits are documented starting from the high-order bit (bit 7) on the left and continuing to the low-order bit (bit 0) on the right.

Parts of Clause 10 present notional implementation aspects as if they were subject to conformance testing. Where such aspects are not externally observable, any such specifications are strictly hypothetical. Only observable, testable aspects of Clause 10 are normative.

10.2 NL functionality overview

10.2.1 General

The NL in this standard performs the following functions:

- Addressing: An NLE determines the appropriate address information for an NPDU.
- Address translation: This standard uses primarily two types of addresses, short DL16Addresses, and long IPv6Addresses. The short DL16Addresses are used within a D-subnet to conserve energy and bandwidth. ALEs, TLEs and NLEs on backbone networks use long IPv6Addresses. The NLE is responsible for translation between the various types of addresses, e.g., when an NPDU moves from a D-subnet to a backbone network (or vice versa).
- NPDU formats: This standard allows for more than one NPDU format to accommodate conservation of energy and bandwidth (which favors short headers), a variety of network topologies, and internetworking with backbone networks. The NLE selects an appropriate format for the NPDU based on such considerations as addressing, routing, level of service, etc.
- Fragmentation and reassembly: NPDU fragmentation and reassembly occurs within the NLE. An NPDU of a size of more than the maximum DSDU size is fragmented by the sending NLE at the point of ingress into a D-subnet. Reassembly is performed by the receiving NLE at the point of egress from the D-subnet.

- Routing: This standard performs routing at two levels: within the backbone network and within the mesh D-subnet. Responsibility for routing at the NL and DL protocol layers is the responsibility of the respective layer entities.

10.2.2 Addressing

ALEs and TLEs in this standard use IPv6Addresses. Each NLE shall have an IPv6Address. If the NLE does not have an IPv6Address prior to the network joining process, the NLE shall be assigned such an IPv6Address by the system manager during the joining process. The NL uses these IPv6Addresses, but does not associate any further meaning to them.

Each NLE compliant with this standard shall also have an IPv6Address that is autoconfigured by the NLE as part of the initialization of its protocol stack. This IPv6Address is referred to as the NLE's link-local address and is derived from the associated DLE's EUI64Address. The format of this IPv6Address is that of a link-local unicast address, as defined in IETF RFC 4291:2006, 2.5.6. Table 200 illustrates this address structure.

Table 200 – Link-local address structure

10 bits	54 bits	64 bits
11 1111 1010	0	EUI64Address

When DPDUs are transmitted over a D-subnet, conveyance of IPv6Addresses consumes valuable bandwidth and device energy resources. Thus this standard defines 16-bit D-aliases for IPv6Addresses so that the short D-aliases are used over the D-subnet. For each D-subnet, a unique DL16Address shall be assigned to each DLE within that D-subnet, as well as to each DLE outside the D-subnet with which a DLE within the D-subnet has a contract. This allows short D-addresses to be used in the D-subnet to represent all origin and destination NLEs.

The scope of a DL16Address is the D-subnet within which it has been defined. Thus, a particular device may have one D-address in the D-subnet to which it belongs and a different D-address in a foreign D-subnet. When a DL16Address is used, it is carried in the DPDU's header.

During the joining process, an NLE might not yet have an IPv6Address and its associated DLE might not have a DL16Address. In this case, TPDUs between the joining device and the advertising D-router shall use the link-local IPv6Addresses when needed (e.g., for the TPDU pseudo-header in join TPDUs). The joining device and the advertising router shall be identified as such by using their EUI64Addresses in the DPDU headers that convey the join messaging.

The system manager assigns the DL16Address and IPv6Address of each DLE and NLE, respectively, that operate in a WISN conforming to this standard. These addresses are assigned during the subnet joining process. The NLE specified by this standard supports only unicast addressing.

NOTE 1 Multicast addressing is a subject for future standardization.

NOTE 2 Backbone and plant network technologies are outside the scope of this standard. Therefore this standard does not specify the representation of IPv6Addresses on a particular backbone or plant network.

10.2.3 Address translation

Since this standard employs DL16Addresses within a D-subnet, when an NPDU moves from a D-subnet to a backbone network (or vice versa), the NLE of the backbone router shall translate between the DL16Addresses and the IPv6Addresses. The same kind of translation shall be performed by the NLE of a D-subnet endpoint.

All devices in this standard shall maintain an address translation table (ATT), as shown in Table 201.

Table 201 – Address translation table (ATT)

D-address (DL16Address)	N-address (IPv6Address)
N1_16	N1_128
N2_16	N2_128
GW_16	GW_128
BBR_16	BBR_128
SM_16	SM_128

The address translation table is initialized during the subnet joining process with the DL16Address and the IPv6Address of the system manager. This information is part of the non-security component of join response, received from the system manager as described in 6.3.9.2.

The address translation table shall be updated by the source NLE whenever a communication session is established with a new destination NLE. Communication sessions are described in 6.3.11.2.5.2. The DL16Address and the IPv6Address of the destination NLE and its associated DLE are stored in the address translation table upon the successful completion of the session establishment process. The process of session establishment is described in 7.5. If a session is terminated for whatever reason, any entry associated with the destination device shall be deleted.

An NLE maintains entries in its ATT for other NLEs with which it communicates; these other NLEs may either belong to the same D-subnet as the first NLE or have a DL16Address in the same D-subnet.

Within a particular D-subnet, an NLE (whether local or remote) shall have only one DL16Address. Thus, the ATT can be used for both forward and reverse lookup by the NLE. For example, an ATT_lookup function can be defined which takes the address to be looked-up as a parameter:

- IPv6Address determined through ATT_lookup of a DL16Address;
- DL16Address determined through ATT_lookup of an IPv6Address.

It is possible to package multiple NLEs or DLEs in a single physical device to support multi-homing. Although such operation is not specified by the standard, it is not prohibited.

An address with no entry in the ATT shall be translated with the help of the system manager. For each NLE joining the network, the system manager shall maintain the IPv6Address of the NLE and the associated DL16Address or D-alias for each D-subnet in which the NLE has such an alias. Hence, the local ATT at an NLE shall be updated through the system manager.

The ATT is an integral part of the NLMO and can be directly updated by the system manager by using the NLMO manipulation methods described in Table 210.

If a lookup in the ATT yields no results, then the lookup function notifies the NLMO. The NLMO issues a read primitive to the directory service object (DSO) in the system manager to obtain the appropriate translation. The lookup function returns a value of null if the system manager also has no mapping for a particular address or the system manager is not available. Any new information from the system manager is stored in the ATT table.

This process is illustrated in Figure 91.

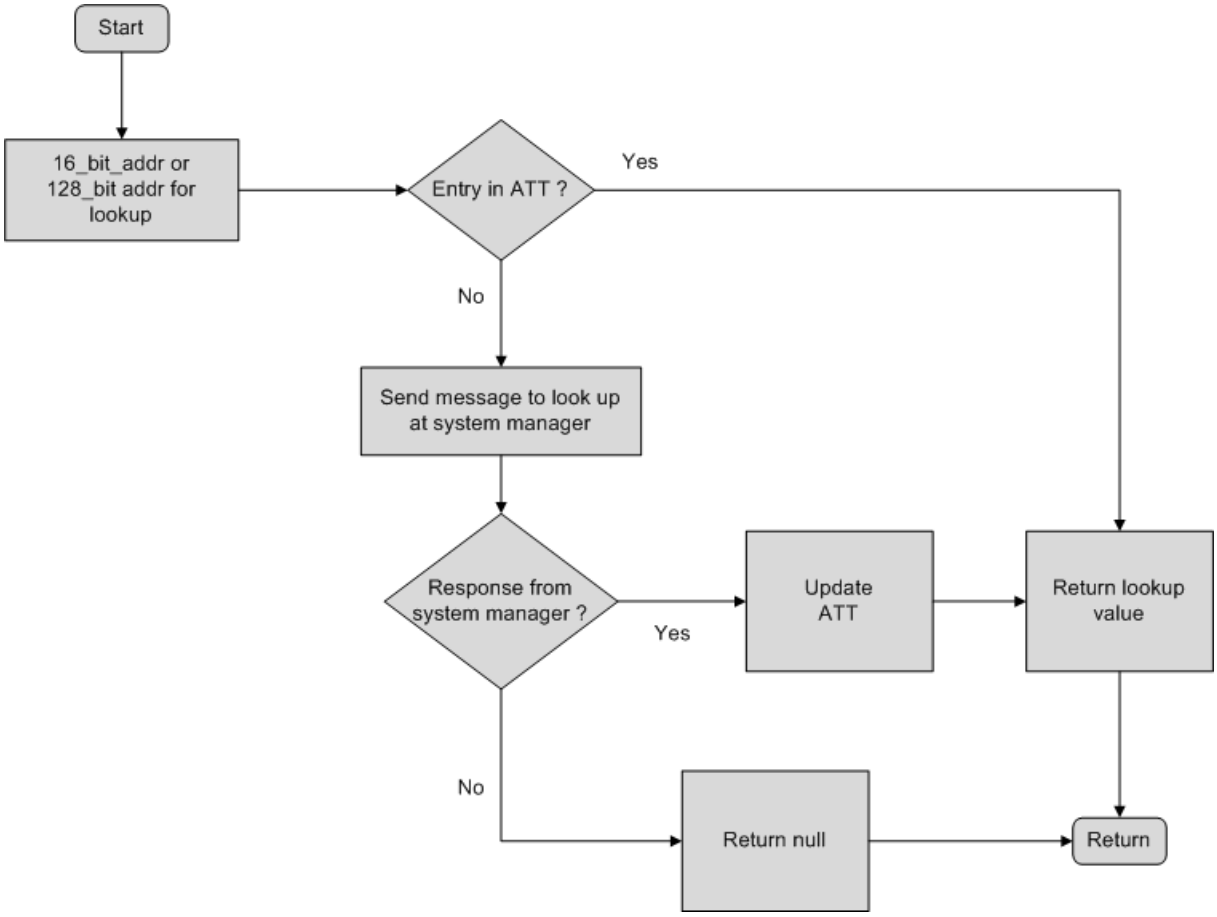


Figure 91 – Address translation process

10.2.4 Network protocol data unit headers

Three formats are used for NPDU headers. The value of the header's first octet provides the means to distinguish between these formats:

- Basic header: This format is intended for NPDUs traversing a single D-subnet and shall be used only over that D-subnet. It is expected to be the most common format in use because its use minimizes the overhead associated with the transmission of headers. The basic header is just an abbreviation for a specific fixed value of the 6LoWPAN compressed header; this value indicates that the source and destination addresses are elided, instead of being conveyed in the DPDU header, as described in 10.5.2.
- Contract-enabled header: This format also is used only over a single D-subnet, when the originating device needs to include more information in the NPDU, such as a contractID. This additional information allows backbone routers to select appropriate resources (e.g., graphID, priority) for the routing of the NPDU, as described in 10.5.3.
- Full header: This is a full IPv6 header, suitable for use over the backbone. NPDUs containing a basic or contract-enabled header shall be expanded into the full header format before routing over the backbone. In return, backbone routers convert full headers into basic or contract-enabled headers for transmission over a D-subnet, as described in 10.5.4.

10.2.5 Fragmentation and reassembly

If the entire NPDU is smaller than the maximum DSDU size, the NPDU shall not be fragmented and the network header shall not contain a fragmentation header. If the NPDU exceeds the maximum DSDU size, the NPDU shall be fragmented into fragmented NPDUs that do not exceed the D-subnet's maximum DSDU size. Fragmentation shall be performed by

the NLE at the point of ingress into a D-subnet. Reassembly shall be performed by the NLE at the point of egress from a D-subnet.

NOTE Origination by a TLE in a D-subnet-connected device constitutes “a point of ingress” into the D-subnet, and similarly delivery to a TLE in a D-subnet-connected device constitutes “a point of egress” from the D-subnet.

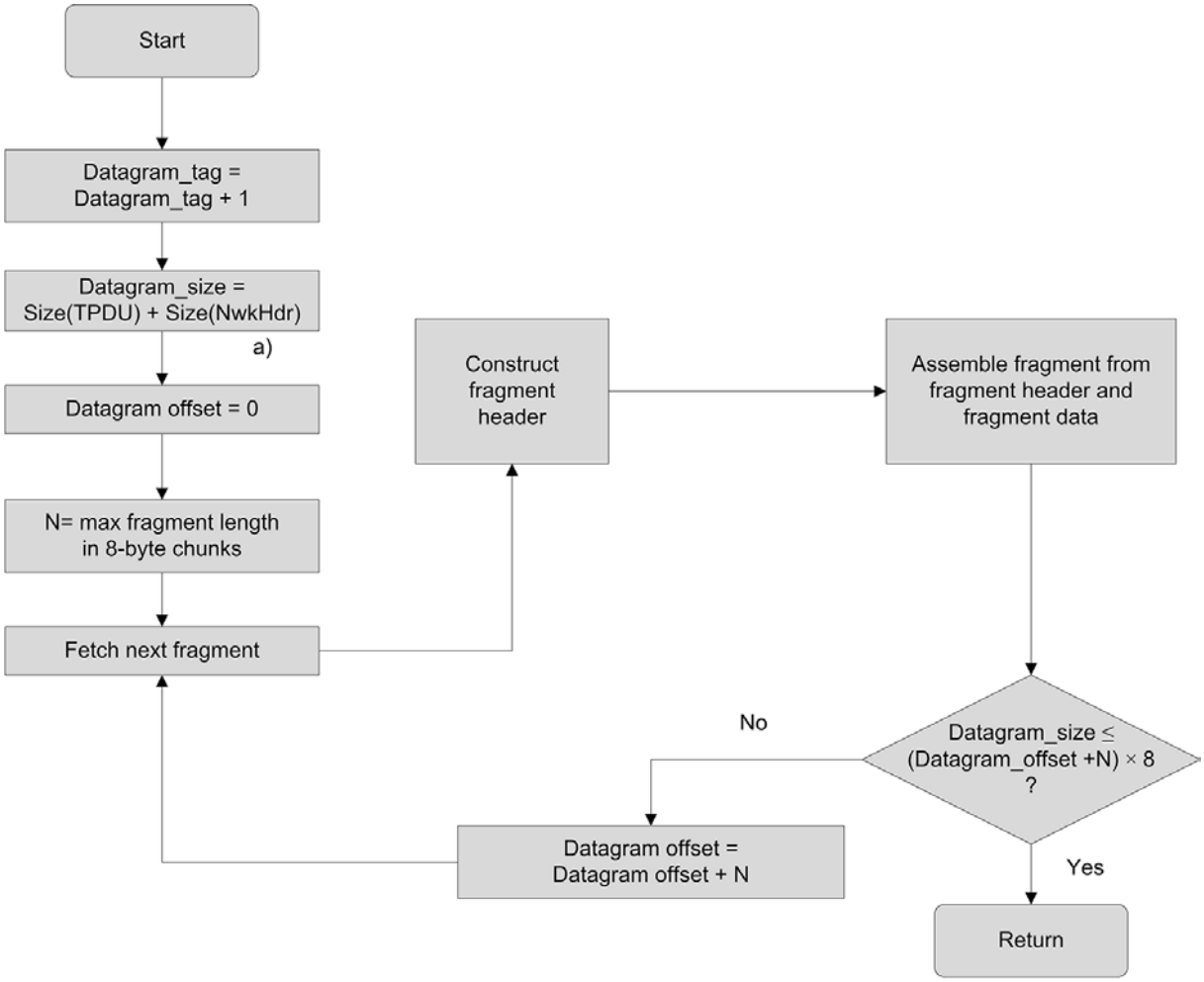
The first fragment shall contain the first fragment header as defined in Table 219. The second and subsequent fragments (up to and including the last fragment) shall contain a fragmentation header as defined in Table 220. The offset of this fragment, referred to as the datagram offset, shall be expressed in units of eight octets, so that each fragment other than the last consists of a multiple of eight octets.

The Datagram_size field shall be present in every fragment, to simplify the reassembly tasks when fragments arrive out of order at their reassembling NLE. The inclusion of the Datagram_size in every fragment allows the receiver to allocate the appropriate amount of buffer space when the first fragment is delayed.

All fragments (first and subsequent fragments) shall have a Datagram_tag field in their header. The value of this field shall be assigned by the device performing the fragmentation and shall be the same for all fragments of the NPDU, so that the reassembling device can recognize that the fragments belong to the same NPDU. To the extent possible, the NLE performing the fragmentation shall assign a different Datagram_tag value to each distinct NPDU that it fragments. To achieve this, each NLE shall have a counter that is initialized to a uniform-random value and is incremented for each NPDU that undergoes fragmentation; the value of this counter shall be placed in the Datagram_tag field of each fragment of the NPDU.

In the extremely rare case that two NPDUs from the same source to the same destination are fragmented by different intermediate routers that coincidentally pick the exact same Datagram_tag, the reassembling device may not be able to disambiguate fragments. In this case, the GraphID may be used to disambiguate further; however, this is not specified as mandatory in this standard. TPDUs reassembled in error from multiple sources will be dropped due to checksum errors and retransmitted. Intermediate routers that fragment NPDUs may also coordinate their fragmentation state machines in order to avoid scenarios in which the reassembling device might not be able to disambiguate fragments.

Figure 92 illustrates the fragmentation process.



a) This is the size of the NwkHdr excluding any contained fragmentation subheader.

Figure 92 – Fragmentation process

To identify all fragments that belong to the same NPDU, the reassembling NLE shall use:

- the source IPv6Address;
- the destination IPv6Address;
- the datagram_tag; and
- the datagram_size.

Otherwise the NLE shall begin reconstructing the original unfragmented NPDU, whose size is Datagram_size, using the Datagram_offset field to determine the relative location of the individual fragments within the original unfragmented NPDU.

When an NLE first receives a fragment with a given Datagram_tag that requires reconstruction, it starts a reassembly timer. If this timer expires before the entire NPDU has been reassembled, the received fragments shall be discarded. The reassembly timeout shall be set to a value defined in NLMO.Frag_Reassembly_Timeout (attribute identifier 11 in Table 206). If a fragment that partially overlaps another fragment is received, and it differs in either the size or Datagram_offset of the overlapped fragment, the fragment(s) already accumulated in the reassembly buffer shall be discarded.

The text just before Figure 92 provides one example of how such inconsistent fragmentation can arise.

A new reassembly commences with a fragment containing a tag for which no fragments are pending. This may lead to buffers being allocated when some fragments arrive after the timeout of the reassembly process that had been previously initiated for the same tag (in essence, attempting to reassemble the NPDU a second or later time). That repeated reassembly usually will fail to complete, causing the new buffers to eventually be flushed due to a NLMO.Frag_Reassembly_Timeout.

The reassembly process is completed when the NPDU is fully reassembled or the timer expires. If the NPDU exceeds the size indicated by NLMO.Max_NSDU_size, the reassembly process may be aborted and the NPDU may be discarded. The device may send a dropped PDU/PDU error alert with value 7 indicating that it is out of memory. Dropped PDU/PDU error alerts are shown in Table 211.

The NPDU reassembly process is shown in Figure 93.

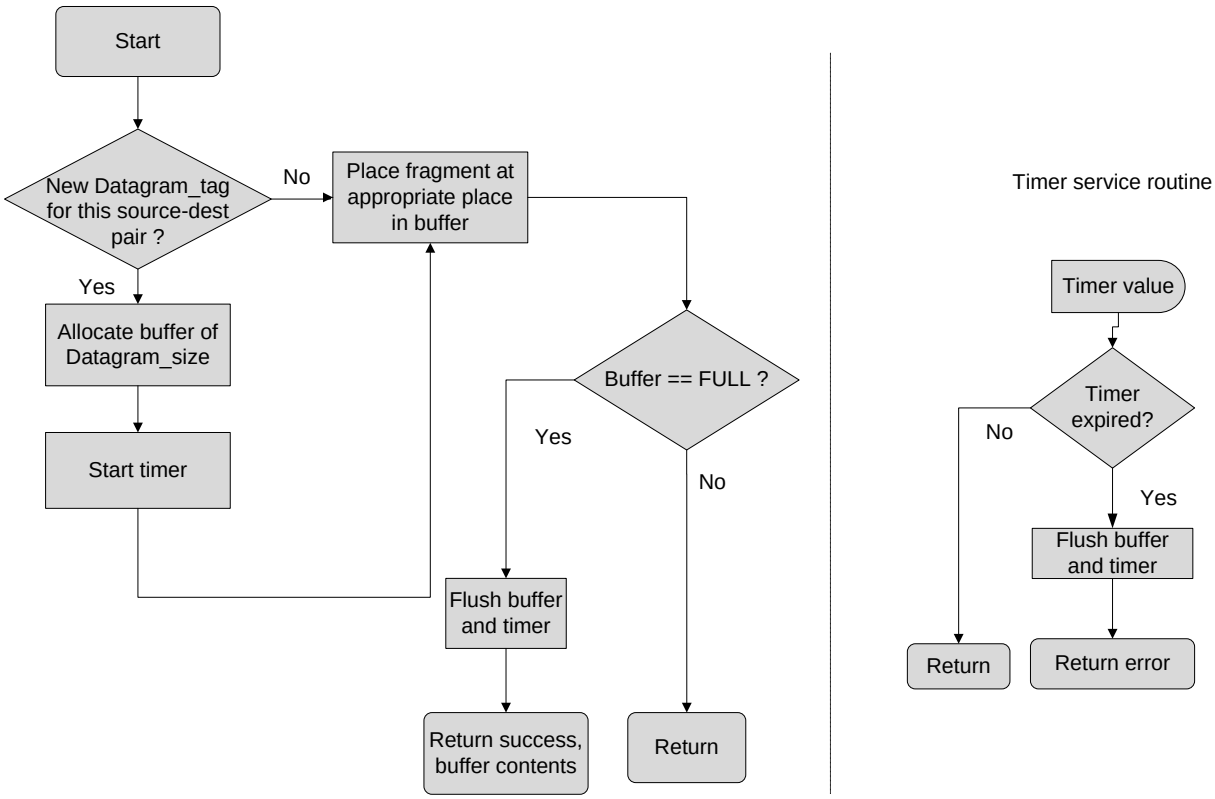


Figure 93 – Reassembly process

10.2.6 Routing

10.2.6.1 General

Routing within a network compliant with this standard happens at two levels:

- One level comprises the endpoints and the backbone devices, if any; the NL is responsible for routing PDUs at this level. This level does not handle routing over the DL links; traversal of a D-subnet appears as a single hop to an NLE.
- The second level of routing is within a D-subnet. This level is the responsibility of the DL (a layer 2 mesh implementation).

The routing between D-subnets and backbone networks is the responsibility of the NL, whose NPDUs conform to the IETF IPv6 and 6LoWPAN standards. This standard specifies minimum requirements for routing, along with notional management services for adding, deleting, and maintaining routes.

10.2.6.2 Routing tables

The NLE in devices compliant with this standard shall maintain a routing table (RT) to keep track of the next hop for a given destination. This table shall be maintained using IPv6Addresses, since such addresses are unique across an entire network compliant with this standard (including all D-subnets). An example of a routing table is provided in Table 202. The routing table may be updated at the source device whenever a communication session is established with a destination device.

Table 202 – Example of a routing table

DestinationAddress	NextHop	NWK_Hop_Limit	OutgoingInterface [a]
N1	BBR1	2	Backbone
N2	BBR1	2	Backbone
GW	GW	2	Backbone
N3	N3	1	D-subnet
N4	N4	1	D-subnet
N5	N5	1	D-subnet
…	…	…	…

[a] This field is set to D-subnet for all destinations in routers and I/O devices.

NOTE In this standard, the route table and all NL management objects are specified to support only one active D-subnet at a time. All DL16Addresses are unique within the scope of that single D-subnet. This is not intended to prevent a device from participating in multiple D-subnets simultaneously. Multiple D-subnets are represented by multiple NLEs.

DLEs that are not backbone-capable only route DPDUs within the D-subnet. Routing within the D-subnet is the responsibility of the DL (a layer 2 mesh implementation). Hence DLEs that operate in the D-subnet but are not backbone capable may maintain a routing table but are not required to do so. This is also reflected in Table B.18 that normatively presents the minimum routing table sizes that need to be supported by devices that meet various role profiles. NL routing tables provide layer independence and allow potential route-over implementations, where routing within the D-subnet is achieved through NL routing.

The routing table shall also be used by the backbone routers to decide whether to route a PDU over the backbone or over the D-subnet of this standard. The OutgoingInterface field indicates whether the PDU shall be sent over the backbone or over the D-subnet.

NextHop indicates the next device whose NLE shall process the NPDU destined for the DestinationAddress. Any device reachable through the DL mesh has NextHop equal to the destination address and the NWK_Hop_Limit field set to 1. From the perspective of the NLE, any device that is reachable through the DL mesh is a single network hop away.

10.2.6.3 Processing of a network service data unit received from a TLE

When an NSDU is passed to an NLE by a TLE, the NLE determines the final destination for that NSDU based on the ContractID. The contract table (see Table 207) is used to obtain the destination address. Devices with both a backbone and a DL interface compliant with this standard shall look up the destination address in the routing table to determine which network interface to use. All non-backbone DLEs shall always use their DL interface.

The NLE shall use the ContractTable to obtain the two-bit priority for the contractID in the N-Data.request. This contract priority shall be combined with the two bits of message priority (also passed in the N-Data.request) to obtain a 4-bit NPDU priority that is passed down to the DLE; the two most significant bits shall be the contract priority, and the two least significant bits shall be the message priority. The Discard Eligible (DE) field from the N-Data.request is also passed down to the DLE. If the OutgoingInterface for the destination address is the

backbone then the 4-bit priority and DE eligible bits shall be included in the TrafficClass field of the IPv6 header.

The NLE shall use the ContractTable to check if the ContractID needs to be included in the NPDU. Including the ContractID in the NPDU allows intermediate backbone routers to make appropriate routing choices (level of service, graphID, etc.) on the backbone or a different D-subnet. When routing over the DL interface, if the ContractID does not need to be included then a basic NPDU header should be constructed; otherwise, a contract-enabled NPDU header should be constructed.

The NLE shall also determine whether fragmentation is needed for the NPDU and shall perform the fragmentation process if necessary. Fragmentation shall be required only for NPDUs routed over a D-subnet; dlmo.MaxDSDUSize shall indicate the maximum payload that can be carried over the D-subnet. If the DSDU size is greater than this value, then fragmentation is necessary.

BBR caching mechanisms and inter-BBR forwarding and reassembly protocols can provide the necessary functionality to permit NSDU fragments that arrive (from the D-subnet) at differing BBRs to be reassembled and forwarded by one of those BBRs.

NOTE A future edition of this standard may specify such an inter-BBR mechanism and protocol.

Unless the configuring system manager knows that the selected BBRs have the necessary capability, NPDUs requiring fragmentation shall not use D-subnet routes that terminate in more than one BBR, because the non-initial NPDUs resulting from 6LoWPAN fragmentation do not carry sufficient information for them to be routed directly to their intended final destination on the backbone subnet before reassembly has occurred.

Figure 94 illustrates the processing of an NSDU received from a TLE.

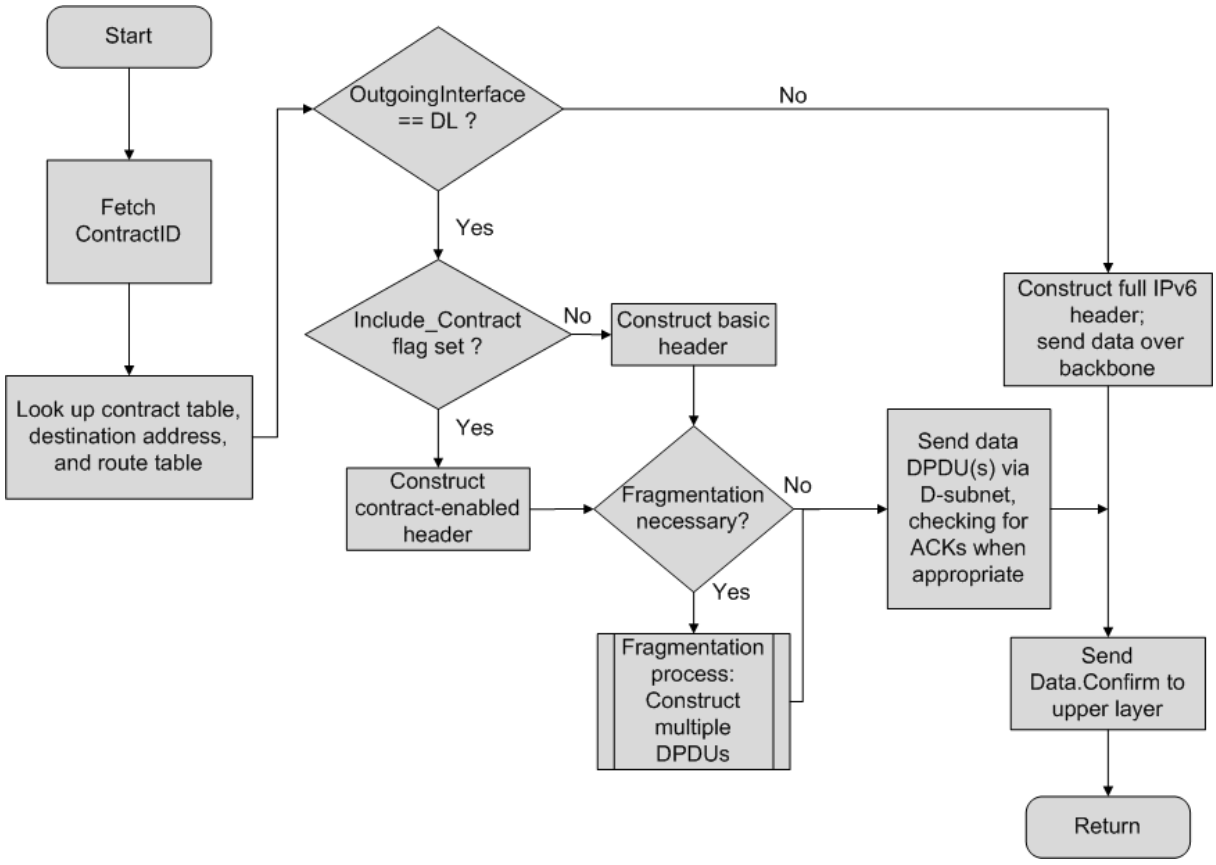
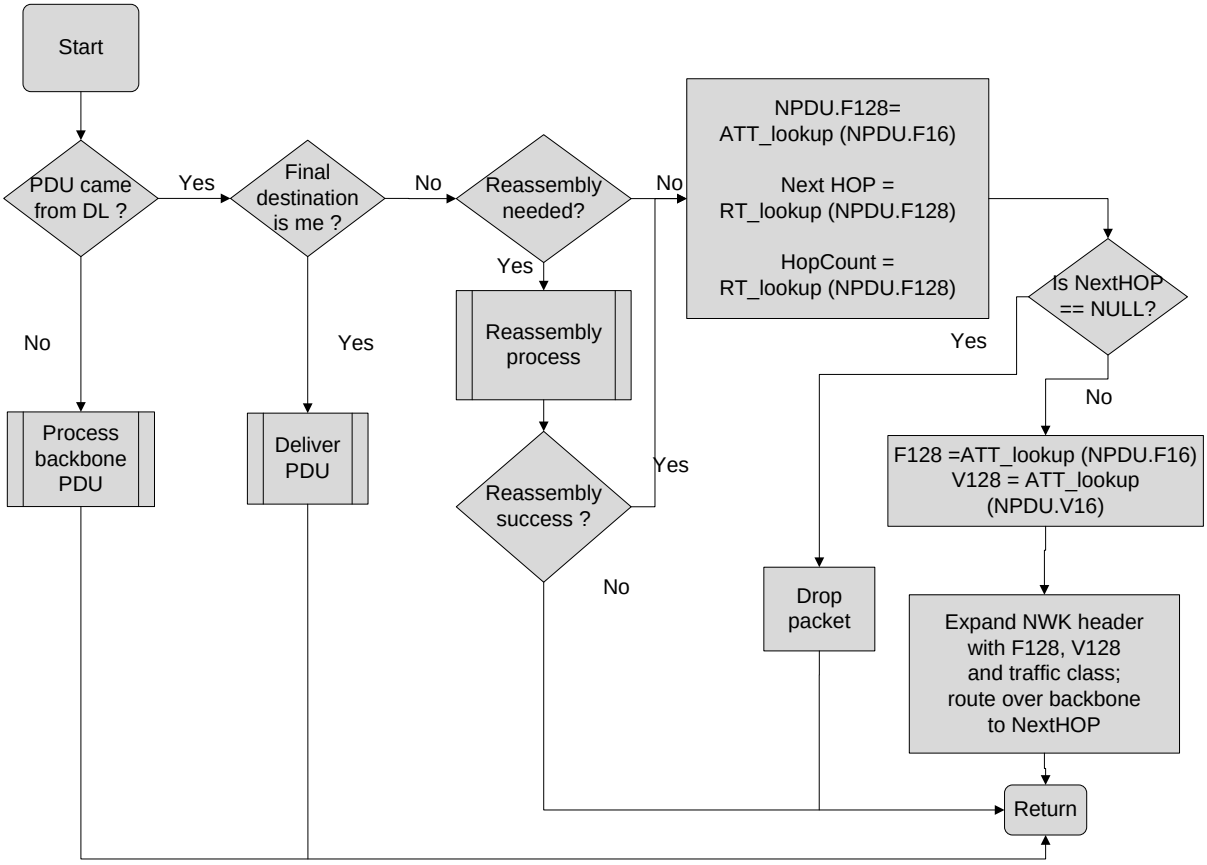


Figure 94 – Processing of an NSDU received from a TLE

10.2.6.4 Processing of a received NPDU

A received NPDU (i.e., a packet), whether received from the DL or the backbone interface, shall first be checked to determine if the final destination is a TLE of the current device. If so, the NSDU that is conveyed as the payload of the NPDU (after any required NPDU defragmentation) shall be passed to the collocated TLE. If the final destination is not the current device, then the device shall route the NPDU appropriately (via either the backbone or the associated DLE). The overall decision process is shown in Figure 95. Not all packets received from the DLE will have a corresponding DL16Address entry in the ATT. Some devices operating on the backbone may not have an assigned DL16Address, but only an IPv6Address. In such a case the NPDU will be delivered by the local DLE after being forwarded from the sending remote DLE over a default route. In that case the backbone-capable device will directly look up the route associated with the destination IPv6Address.



NOTE BBR coordination of packet reassembly of NPDUs that were fragmented by 6LoWPAN for transmission across the very limited payload capacity D subnet is supplier-specific and beyond the scope of this standard.

Figure 95 – Processing of a received NPDU

The DLE's notional DD-DATA.indication and DD-DATA.request services convey a LastHop (LH) parameter. When this LH parameter is set, it indicates that the PDU entered the D-subnet through a backbone router, and therefore is prohibited from exiting the D-subnet through a backbone router, thus avoiding circular routes within the NL. This restriction enables the NLE to elide the Hop Limit field from a compressed NPDU that uses the basic header format while still preventing circular routing.

When the NPDU is received from the DL at a device other than the destination, if the LastHop (LH) parameter is set in the DD-DATA.indication, the NPDU has reached the current device in error and shall be discarded. If the NPDU is received from the D-subnet and not discarded (see Figure 95), the intermediate router shall first fully expand the NPDU's network header.

As part of this expansion, the explicit congestion notification (ECN) value provided by the DD-DATA.indication shall be included in the appropriate field of the expanded header.

After any header expansion, the receiving DLE shall check to see whether reassembly (due to prior fragmentation) is needed for this NPDU. Once any needed reassembly completes, the NPDU shall be prepared for routing over the backbone. The DL16Address of the origin (very first V) and destination (final destination F) in the DD-DATA.indication shall be translated into IPv6Addresses. Then the routing table shall be used to determine the next NL hop for reaching the final destination. The NPDU shall be presented to the NLE of the backbone interface for routing on the backbone network. This standard does not specify how the backbone handles and routes the NPDU. The backbone has the responsibility to deliver the NPDU to the NLE of the NextHop.

This standard always uses ECN. When congestion notification is carried in a DPDU header, if the ECN bits are non-zero in the NPDU header, they shall be set to zero in that header to indicate that the notification is carried in the DPDU header. A backbone router NLE that receives a potentially-reassembled NPDU from its associated DLE shall use the ECN information carried in the received DPDU header to fill in the ECN bits in the expanded NPDU header. NPDUs originating from backbone devices shall have the ECN bits set to indicate that explicit congestion notification is used.

If an NPDU is received from the backbone, it will have an expanded header, and the final destination and very first (originator) addresses will already be expressed as IPv6Addresses. If the NPDU needs to be routed over a D-subnet, the DL16Addresses in that D-subnet of the very first (originator) DLE and final destination DLE shall be obtained from the ATT and passed to the DLE in the DD-DATA.request. The NLE shall check if the ContractID and priority are included in the FlowLabel and TrafficClass fields, respectively, of the expanded NPDU header. If so, the ContractID and priority shall also be passed to the DLE to allow the selection of appropriate DL routing mechanisms (GraphID, etc.).

The presence or absence of congestion is determined from the ECN field of the NPDU received from the backbone, which is passed to the local DLE in a DD-DATA.request. When passing an NPDU with a basic header to a local DLE, then the LastHop (LH) parameter shall be set in the DD-DATA.request to indicate that the NPDU has entered a D-subnet from which it is not allowed to exit. If the NPDU size exceeds dlmo.MaxDsduSize for this D-subnet, the NPDU shall be fragmented before conveyance as DSDUs. This process is depicted as a flowchart in Figure 96.

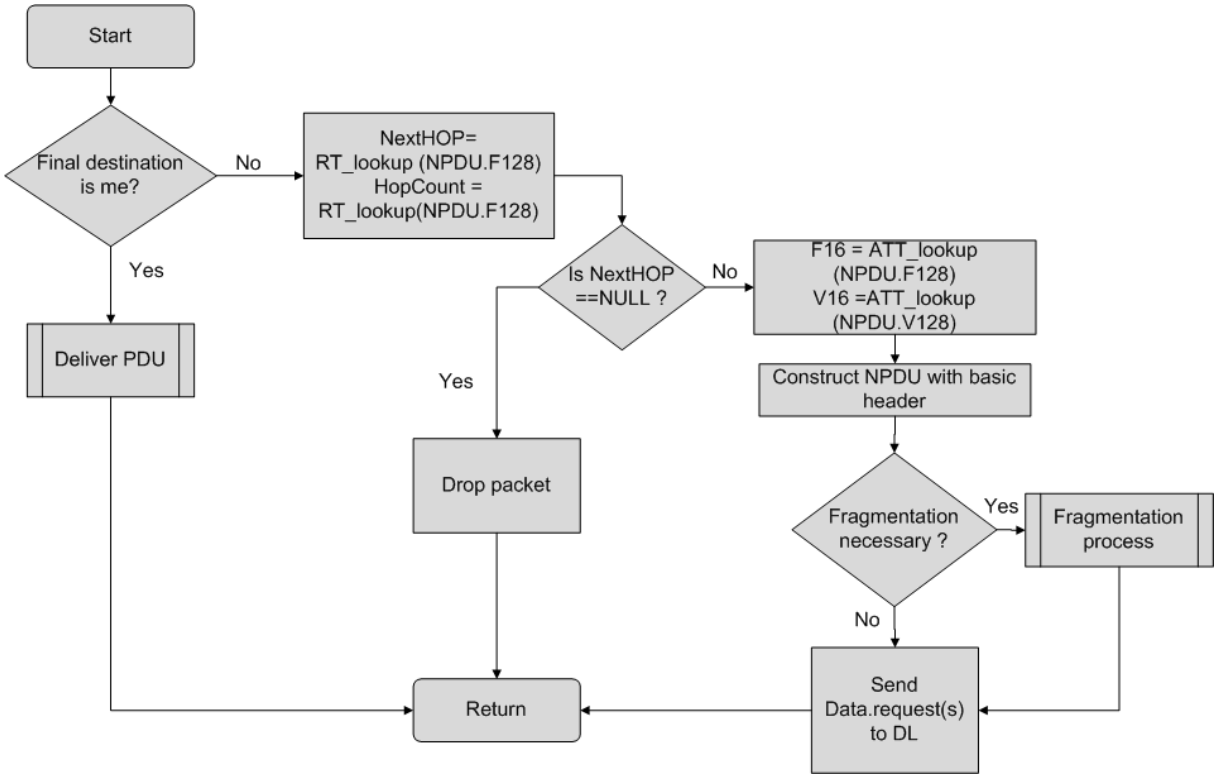


Figure 96 – Processing of a NPDU received by a NLE from the backbone

If the receiving NLE is the intended final destination, then that NLE shall process the NPDU and shall pass the conveyed NSDU up to an associated local TLE, along with an indication of whether congestion was encountered, as conveyed by the NPDU's ECN bits. The NLE shall first check if it has received a fragment; if so, it shall perform the reassembly process (see Figure 93). The NPDU's source IPv6Address shall be translated to an IPv6Address, if necessary, and the NSDU shall be passed to the associated TLE. Figure 97 depicts the flowchart for this processing.

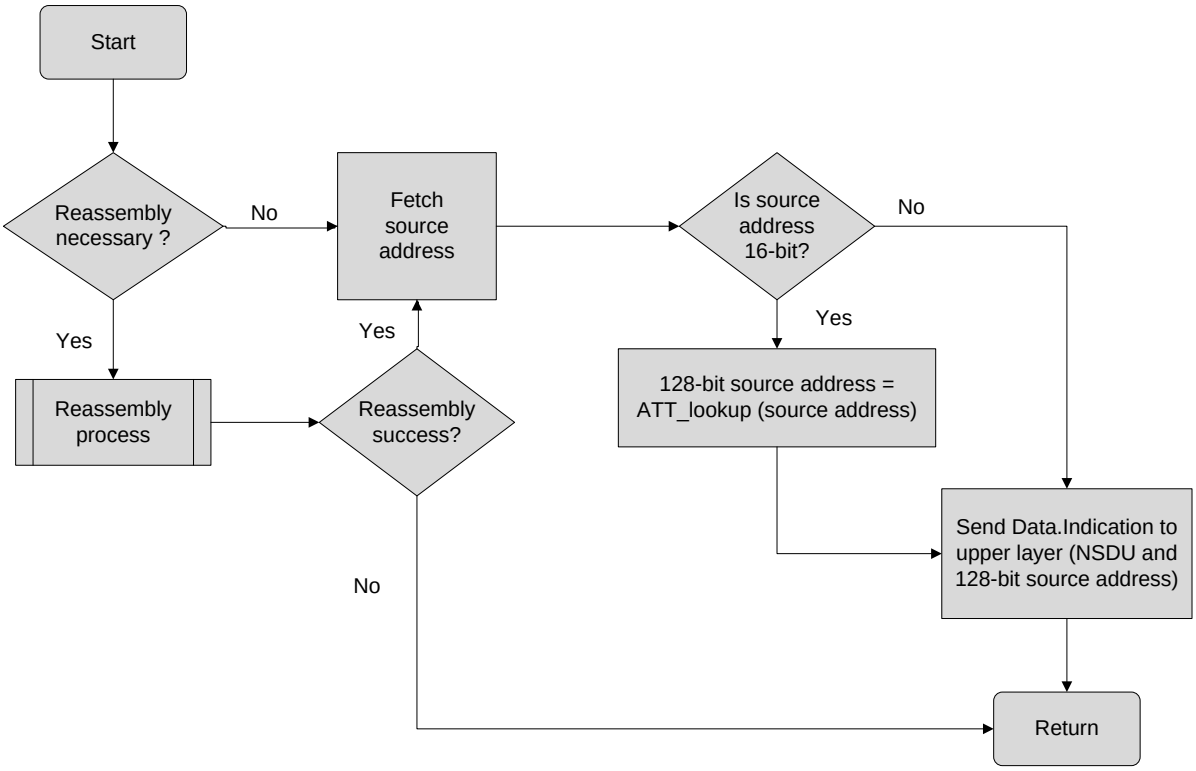


Figure 97 – Delivery of a received NPDU at its final destination NLE

10.2.7 Routing examples

10.2.7.1 Routing from a field device direct to a field-connected gateway

Figure 98 illustrates routing from a field device to a gateway with no backbone routing.

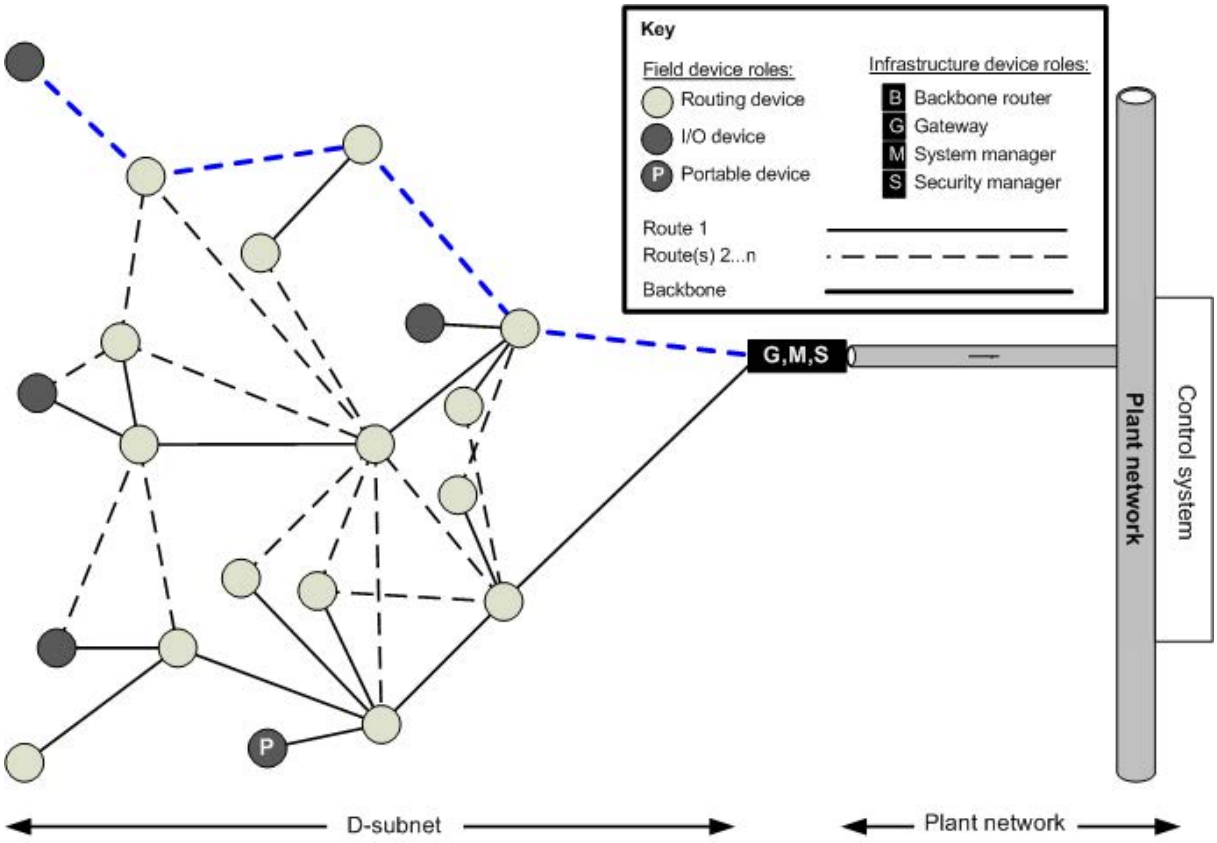


Figure 98 – Routing from a field device direct to a field-connected gateway without backbone routing

Figure 99 depicts the routing path through the communication protocol suites as the PDU moves from an I/O device to the gateway. It is assumed that the NPDU needs no fragmentation.

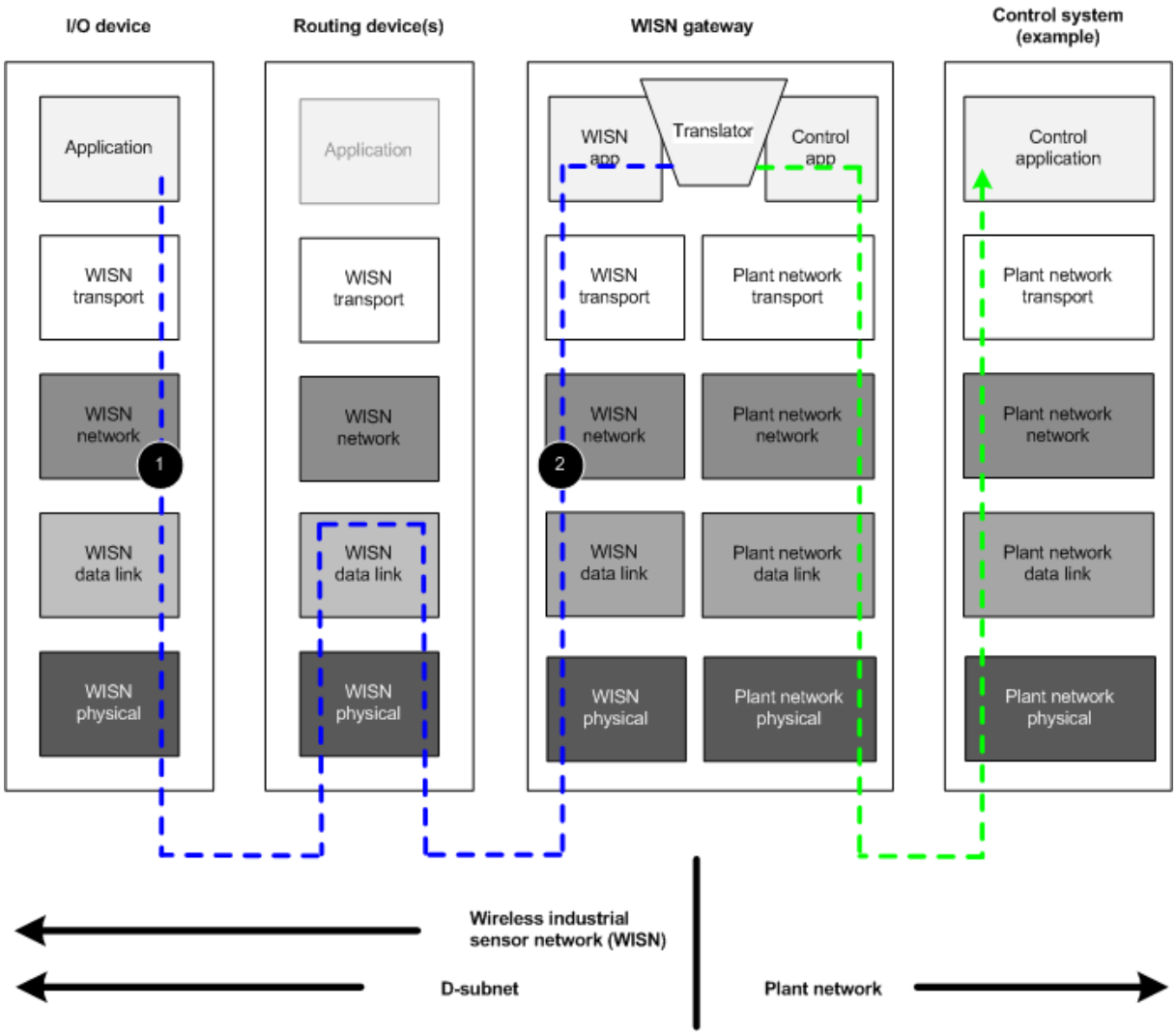


Figure 99 – Protocol suite diagram for routing from a field device direct to a field-connected gateway without backbone routing

In Figure 99, the gateway is shown to have a field medium; hence no backbone network is involved in this example. The blue dotted lines (on the left) depict messaging flow on the network on the low-side (i.e., field side) interface of the gateway, while the green dotted lines (on the right) depict the messaging flow on the network on the high-side (i.e., operational management side) interface of the gateway.

The operations of the NLEs at the devices that the NPDU traverses (numbered in order) are as follows:

a) The NLE in the originating field device uses a basic network header; the DL16Address of the gateway and the DL16Address of the device itself are obtained from the ATT and passed to the DLE as a DSDU for conveyance to the gateway.

b) The NLE in the gateway receives the NPDU, checks that the NPDU is intended for the gateway, translates the DL16Address of the originating device (provided by the DD-DATA.indication) into an IPv6Address, and then passes the NSDU to the TLE.

10.2.7.2 Routing from a field device to a gateway via a backbone router

Figure 100 illustrates the routing of a PDU from a field device to a gateway via a backbone router.

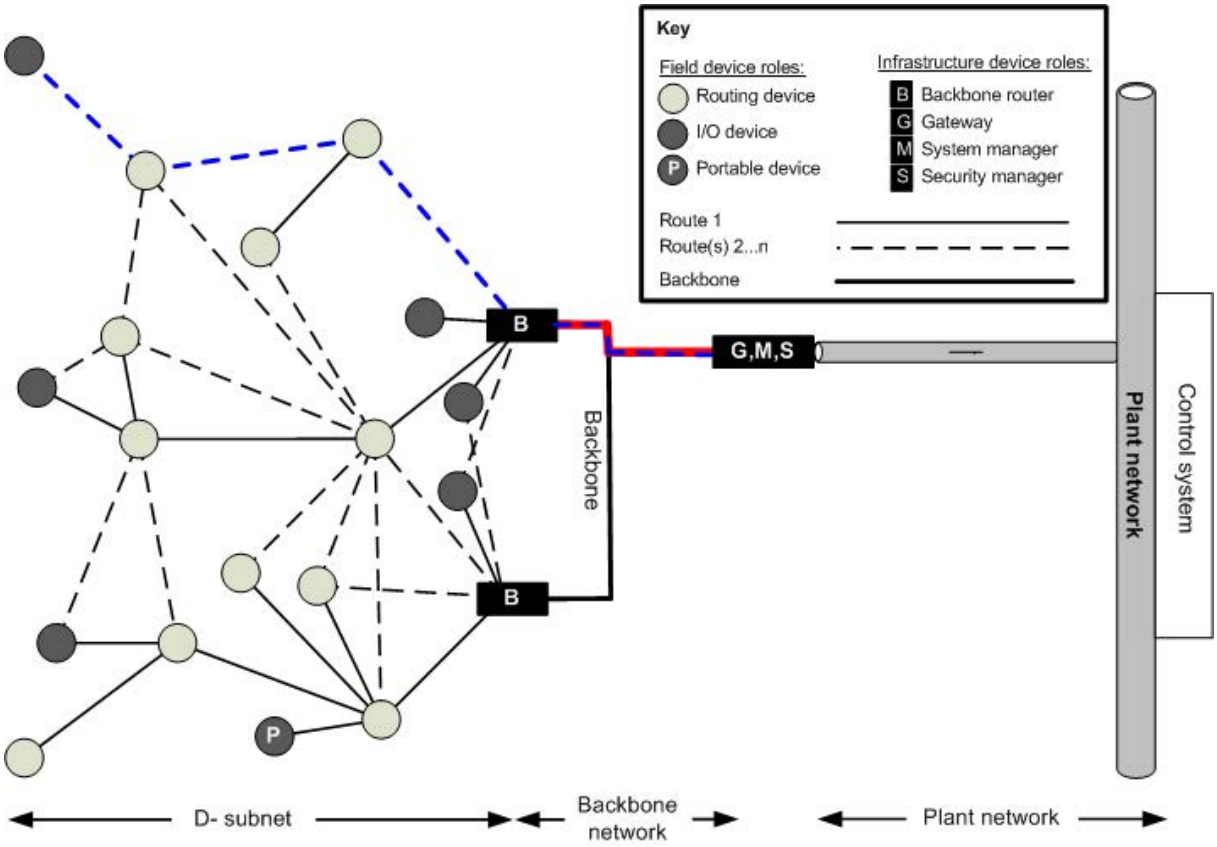


Figure 100 – Routing an NPDU from a field device to a gateway via a backbone router

Figure 101 depicts the flow of an NPDU from a field device to a gateway resident on the backbone network.

The NPDU is first routed to a backbone router over the D-subnet, and from there to the gateway over the backbone. The operation of the NLEs at the devices that the NPDU traverses (listed in order) is as follows:

a) The NLE of the I/O device passes to its local DLE its own DL16Address as the source address and the DL16Address of the gateway as the final destination address. If the ContractTable indicates that the ContractID needs to be included in the NPDU, the contract-enabled header is used; otherwise, the basic header is used if the compression used by the transport allows it (see 10.5.2.1). If the size of the NPDU is larger than maxDSDU size, the NPDU is fragmented. The complete NPDU (or the set of fragment NPDUs) is then passed as DSDU(s) to the associated DLE.

b) The DLE conveys the DSDU to the backbone router. If fragmented in a), the set of fragments is reassembled as the NPDU at the backbone router. The NLE at the backbone router receives the NPDU and determines that the NPDU is not intended for the backbone router, since the final destination address in the DD-Data.indication is the DL16Address of the gateway. The backbone router translates this DL16Address into the IPv6Address of the desired gateway, uses its routing table to determine the next-hop address to reach the gateway and creates a full header (format shown in Table 216).

c) The reconstituted NPDU with the expanded network header is presented to the backbone interface. The backbone interface routes the NPDU towards its final destination. In this example, the next hop is the final destination (the gateway).

d) The NPDU arrives at the NLE of the gateway over the backbone. The NLE at the gateway determines that the final destination address is equal to the address of the gateway itself and passes the NSDU to its TLE.

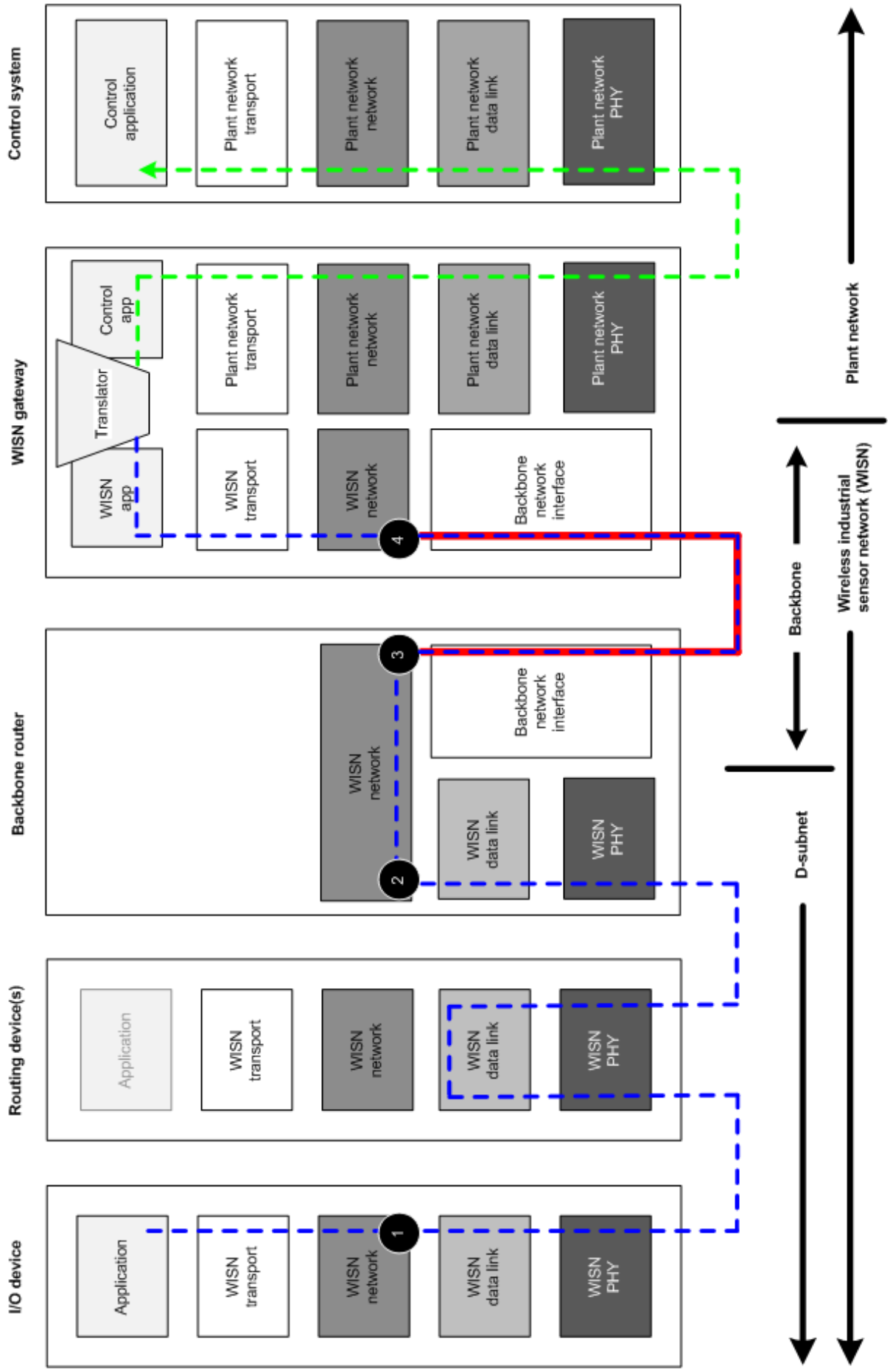


Figure 101 – Protocol suite diagram for routing an APDU from a field device to a gateway via a backbone router

10.2.7.3 Routing from a field device to another field device on a different D-subnet

Figure 102 illustrates routing from an I/O device on one D-subnet to another I/O device on a different D-subnet.

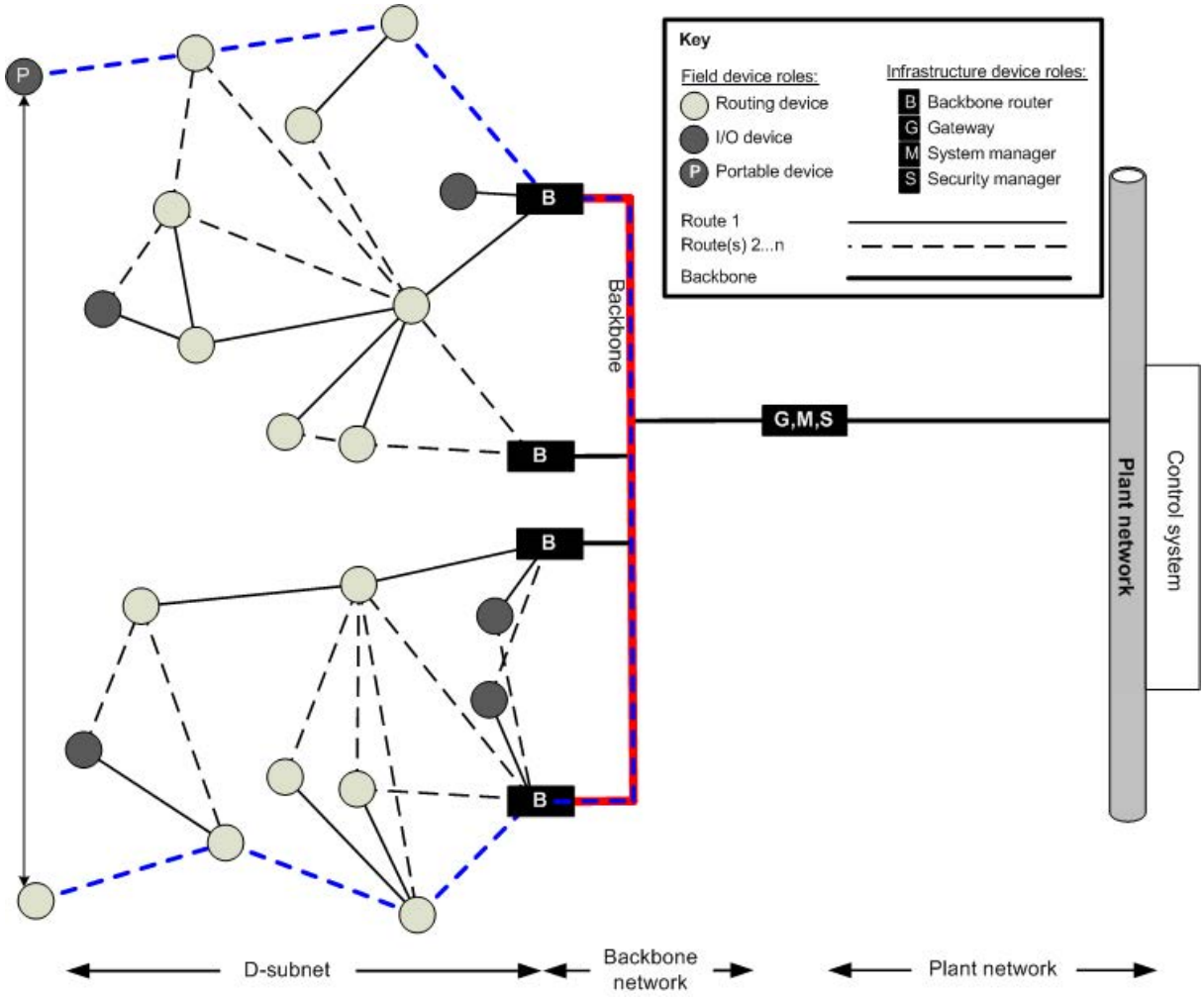


Figure 102 – Routing from a field device on one D-subnet to another field device on a different D-subnet

Figure 103 shows the flow of an NPDU between two field devices on different D-subnets (see 10.2.7.3). It is assumed that the NPDU needs no fragmentation.

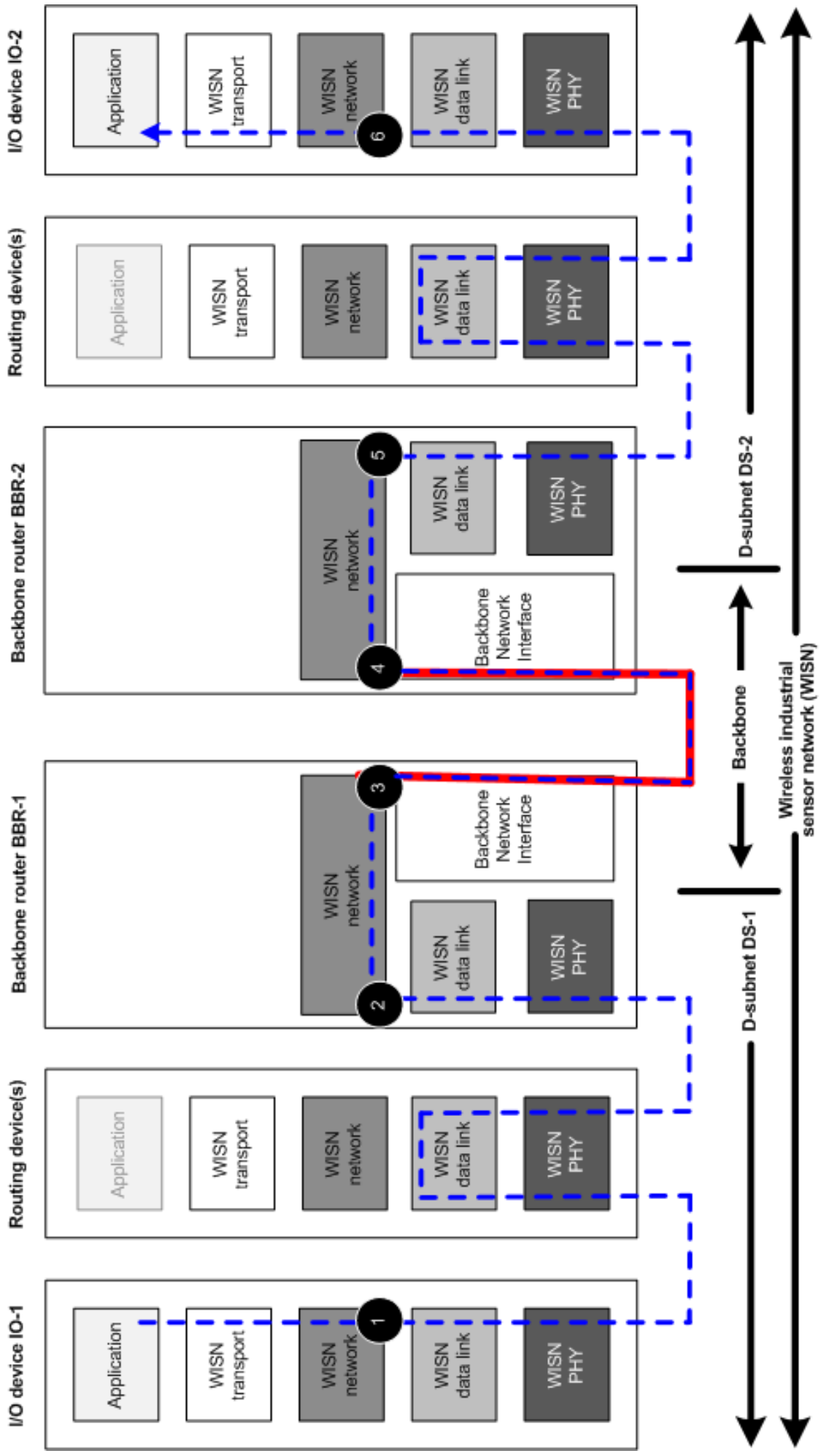


Figure 103 – Protocol suite diagram for routing from an I/O device on one D-subnet to another I/O device on a different D-subnet

The NPDU is routed over the backbone from one D-subnet to the other. The operations performed by the NLEs of the devices as the NPDU moves from the originating I/O device (I/O-1) located in D-subnet DS-1 to the destination I/O device (I/O-2) located in D-subnet DS-2 are as follows:

a) The NLE at I/O-1 creates the NPDU using the contract-enabled network header. The NLE passes the NPDU to the associated DLE as a DSDU, along with the DL16Address of I/O-1 within DS-1 as the source address, and the DL16Address of I/O-2 in DS-1 as the final destination address, via the notional DD-DATA.request. The ContractID is placed in the FlowLabel field of the contract-enabled header.

b) The resulting DPDU(s) is/are routed over DS-1 and arrive(s) at the DLE of BBR-1, i.e., the backbone router in DS-1. The DPDU payloads are used to regenerate the NPDU, which is checked to see if it is destined for BBR-1 itself. Since it is not destined for BBR-1, the DL16Addresses of I/O-1 and I/O-2 in the notional DD-DATA.indication are translated into their IPv6Addresses.

c) The expanded header (in the format defined in Table 216) is created and presented to the backbone interface. The next hop for this NPDU over the backbone is determined by looking up the IPv6Address of the final destination in the RT. The RT returns the IPv6Address of BBR-2 (the backbone router of DS-2) as the next hop, since BBR-2 is the backbone router serving I/O-2. The ContractID is placed in the FlowLabel field of the expanded header. The priority of the PDU is placed in the TrafficClass field.

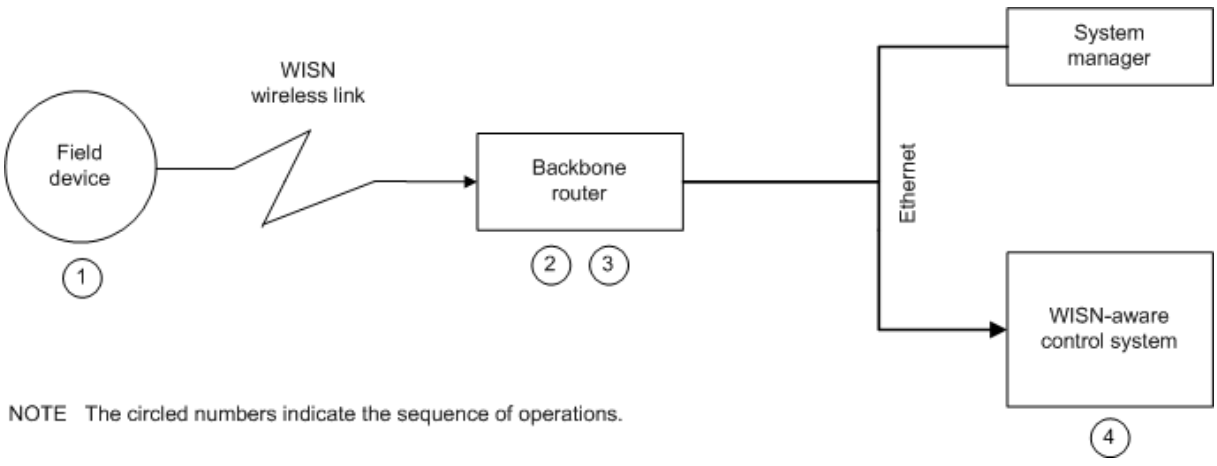
d) The NLE at BBR-2 receives the NPDU over the backbone. The NPDU indicates that the final destination is the IPv6Address of I/O-2. This NPDU is then prepared for routing over DS-2 to reach I/O-2.

e) The NLE at BBR-2 creates a basic NPDU header. In the subsequent notional DD-DATA.request, the DL16Address of I/O-1 in DS-2 is the originator address and the DL16Address of I/O-2 in DS-2 is the final destination. The ContractID, extracted from the FlowLabel field of the expanded header, and the priority, extracted from the TrafficClass, are also passed down to the DL to enable selection of the appropriate routing resources (GraphID, etc).

f) The NPDU arrives at the NLE of I/O-2. Since the final destination indicated in the notional DD-DATA.indication is the DL16Address of I/O-2 in DS-2, the NLE translates the addresses into an IPv6Address and passes the NSDU to the TL of I/O-2.

10.2.7.4 Example of routing over an Ethernet backbone network

Figure 104 is an example of an implementation of the protocol suite diagram illustrated in Figure 103. In this network, a field device communicates with a control system that is aware of the native protocol of this standard; the backbone network in this example is an Ethernet network carrying IPv6 traffic.



NOTE The circled numbers indicate the sequence of operations.

Figure 104 – Example of routing over an Ethernet backbone network

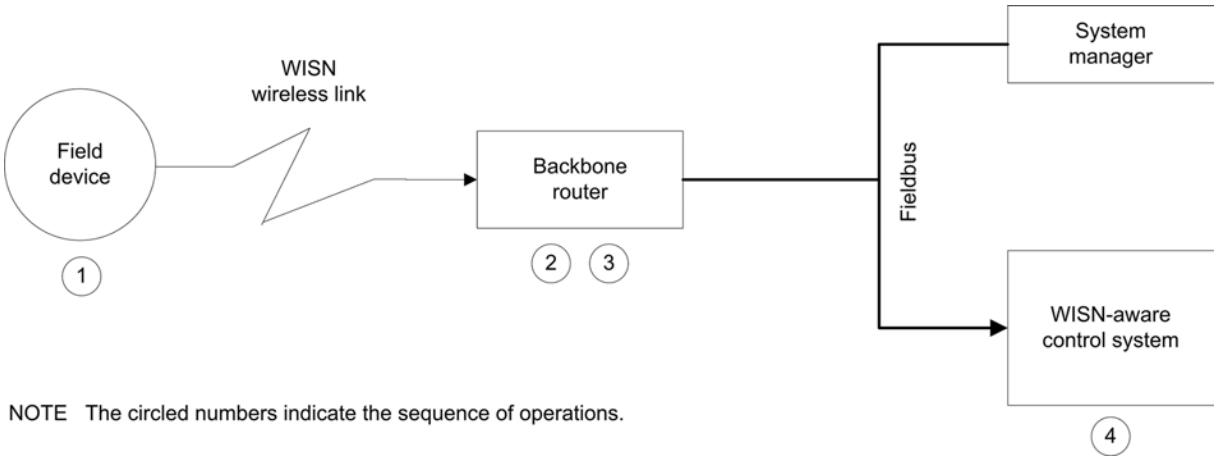
The numbered circles in Figure 104 indicate steps in the routing process and where they occur:

1) The NLE at the field device creates an NPDU and hands it to the associated DLE for transmission to the next NL hop (backbone router). The final destination address of the DPDU is the DL16Address for the native protocol-aware control system. The DL mesh delivers the NPDU to the NLE of the backbone router.
2) The backbone router receives the NPDU, replaces the DL16Addresses with the corresponding IPv6Addresses and expands the NPDU into a full IPv6 NPDU.
3) The backbone router sends the IPv6 NPDU over the Ethernet interface.
4) The NLE at the control system receives the IPv6 NPDU from its Ethernet interface, performs NL processing and passes the NSDU to the TLE.

10.2.7.5 Example of routing over a backbone network

Figure 105 is a variant of Figure 104 that substitutes a generic fieldbus for the Ethernet backbone network. In this variant the IPv6 NPDU is encapsulated for transport over the fieldbus network via one or more fieldbus PDUs.

NOTE Other variants of this fieldbus backbone scenario are possible.



NOTE The circled numbers indicate the sequence of operations.

Figure 105 – Example of routing over a fieldbus backbone network

The numbered circles in Figure 105 indicate steps in the routing process and where they occur:

1) The NLE at the field device creates an NPDU and hands it to the associated DLE for transmission to the next NL hop (backbone router). The final destination address of the DPDU is the DL16Address for the native protocol-aware control system. The DL mesh delivers the NPDU to the NLE of the backbone router.
2) The backbone router receives the NPDU and translates the DL16Address into the IPv6Address and expands the NPDU into a full IPv6 NPDU.
3) The backbone router encapsulates the entire NPDU in one or more fieldbus PDUs addressed to the control system.
4) The control system (gateway) receives the fieldbus PDU(s), extracts the NPDU, performs NL processing, and delivers the NSDU to the associated TLE.

10.3 NLE data services

10.3.1 General

The TLE uses the NLE's NDSAP interface to send and receive data. This interface is internal to a compliant device and is therefore notional and not testable. The internal NSAPs of the device are depicted in Figure 16.

All interfaces between the NLE and its NME or the adjacent TLE and DLE are internal interfaces within the device, and thus are unobservable. Therefore they are not subject to standardization. Thus all of this description is notional.

10.3.2 N-DATA.request

10.3.2.1 General

N-DATA.request is used by a TLE to request the NLE to transmit a TSDU.

10.3.2.2 Semantics

The semantics of the N-DATA.request primitive are as follows:

```
N-DATA.request   (
                 DestAddress,
                 ContractID,
                 Priority,
                 DE,
                 NSDU,
                 NSDUSize,
                 NSDUHandle,
                 ECN
                 )
```

Table 203 specifies the elements for the N-DATA.request primitive.

Table 203 – N-DATA.request elements

Standard data type name: N-DATA.request		
Element name	**Element identifier**	**Element scalar type**
DestAddress (the IPv6Address of the destination NLE for the NSDU)	1	Type: Device address
ContractID (the contract ID associated with the resources to be used for transmitting this NPDU; this ID is passed through directly to the DLE)	2	Type: Unsigned16 Named value: 0 indicates no contract
Priority (priority of the message within the contract) [a]	3	Type: Unsigned2
DE (indicates whether the PDU is eligible for discard)	4	Type: Unsigned1
NSDU (the set of octets forming the NSDU to be transmitted by the NL, including the transport headers)	5	Type: OctetString
NSDUSize (the number of octets in the NSDU to be transmitted)	6	Type: Unsigned16
NDSUHandle (the handle associated with the NSDU to be transmitted)	7	Type: Unsigned16
ECN (explicit congestion notification)	8	Type: Unsigned2
[a] The NLE shall combine this priority with the 2-bit contract priority to encode the D-priority as a 4-bit field before passing it to the DLE.		

10.3.2.3 Appropriate usage

The TLE invokes N-DATA.request to request that the NLE transmit an NSDU.

10.3.2.4 Effect on receipt

On receipt of an N-DATA.request, the NLE constructs the NL headers of the NPDU, first eliding the first octet of the LoWPAN_NHC of the NSDU if the basic header is used, then fragmenting the NPDU (if necessary), and conveying it to the DLE for transmission over the local D-subnet. If ContractID is zero, the NLE treats the NSDU as a join-related PDU with an associated destination EUI64Address, for which the required destination IPv6Address is derived from the link-local address passed as the DestAddress parameter.

10.3.3 N-DATA.confirm

10.3.3.1 General

N-DATA.confirm is used to report the result of an N-DATA.request.

10.3.3.2 Semantics

The semantics of the N-DATA.confirm primitive are as follows:

```
N-Data.confirm   (
                 NSDUHandle,
                 status
                 )
```

Table 204 specifies the elements for the N-DATA.confirm primitive.

Table 204 – N-DATA.confirm elements

Element name	Element identifier	Element scalar type
NSDUHandle (the handle of the NSDU whose status is being reported)	1	Type: Unsigned16
Status (the result of the N-DATA.request primitive that conveyed the NSDU)	2	Type: Unsigned Named value: 0: success

10.3.3.3 When generated

The NLE generates N-DATA.confirm as a delayed response to an N-DATA.request. N-DATA.confirm returns a status to the TLE that indicates either SUCCESS or FAILURE.

10.3.3.4 Appropriate usage

N-DATA.confirm notifies the TLE of the result of its request to transmit an NSDU.

10.3.4 N-DATA.indication

10.3.4.1 General

N-DATA.indication is used by an NLE to deliver a received TSDU to an associated TLE.

10.3.4.2 Semantics

The semantics of the N-DATA.indication primitive are as follows:

```
N-Data.indication (
                  SrcAddress,
                  DestAddress,
                  NSDU,
                  NSDUSize,
                  ECN,
                  Priority
                  )
```

Table 205 specifies the elements for the N-DATA.indication primitive.

Table 205 – N-DATA.indication elements

Element name	Element identifier	Element scalar type
SrcAddress (the IPv6Address of the source of the NSDU)	1	Type: Device address
DestAddress (the IPv6Address of the destination of the NSDU, e.g., the device's own IPv6Address.)	2	Type: Device address
NSDU (the received NSDU, including associated TL headers)	3	Type: Sequence of octets
NSDUSize (the number of octets in the received NSDU)	4	Type : Unsigned16
ECN (explicit congestion notification information from the received NSDU)	5	Type: BitArray4
Priority (4-bit NPDU priority as received)	6	Type: Unsigned4

10.3.4.3 Appropriate usage

The NLE invokes N-DATA.indication to notify the associated TLE of a received NSDU and associated conveyance information. If the received NPDU contains a basic header, then, before passing the NSDU to the TLE, the NLE restores (prepends) the first octet of the NSDU as LoWPAN_NHC (= 0x1111 0111), which had been elided when the basic header was constructed. If the source D-address received from the underlying DLE is an EUI64Address, a link-local IPv6Address is constructed from the EUI64Address and passed as the SrcAddress parameter of the N-DATA.indication. The DestAddress parameter of the N-DATA.indication is the link-local IPv6Address of the device when used for join-related APDUs, or the globally-assigned IPv6Address for post-join operation.

10.3.4.4 Effect on receipt

On receipt of an N-DATA.indication, the TLE is able to process the reported NSDU.

10.4 NL management object

10.4.1 NL management information base

Table 206 specifies the attributes of the NL management object (NLMO).

Table 206 – NLMO attributes *(1 of 3)*

Standard object type name: NL management object (NLMO)				
Standard object type identifier: 123				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute type**	**Description of behavior of attribute**
Backbone_Capable	1	A Boolean flag indicating whether the device is backbone capable	Type: Boolean1 Classification: Static Accessibility: Read only	Fixed value based on device capabilities and implementation details. Backbone capability may be ignored by a system manager
DL_Capable	2	A Boolean flag indicating whether the device is capable of communicating over the wireless Type A medium of this standard	Type: Boolean1 Classification: Static Accessibility: Read only	Fixed value based on device capabilities and implementation details. DL interface capability may be ignored by a system manager
DL16Address	3	The DL16Address of the device	Type: DL16Address Classification: Static Accessibility: Read/write Valid range: 1.. 2^{15} -1	A fixed value as assigned by the system manager at join.[a] This attribute is a duplicate of the corresponding attributes in the DMO and DLMO
Long_Address	4	The IPv6Address of the device	Type: IPv6Address Classification: Static Accessibility: Read/write Valid range: all with high-order bit reset	A fixed value as assigned by the system manager at join.[a] This attribute is a duplicate of the corresponding attributes in the DMO and DLMO
Route_Table	5	The routing table that includes information to route an NPDU	Type: Array of NLRouteTbl structures (see NLRouteTbl) Classification: Static Accessibility: Read/write Valid range: See Table 208	The routing table consists of a destination address, next network hop for that destination and the number of network hops needed. The routing table structure, its corresponding alerts and methods are discussed in 10.2.6.2. The size of the routing table present in devices that only operate within the D-subnet and are not backbone capable is presented in Table B.18
Enable_Default_Route	6	A Boolean value set to indicate if a default routing is enabled	Type: Boolean1 Classification: Static Accessibility: Read/write Default value: Disabled	Enables a default route

Table 206 *(2 of 3)*

Standard object type name: NL management object (NLMO()				
Standard object type identifier: 123				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute type**	**Description of behavior of attribute**
Default_Route_Entry	7	Destination address associated with the default route	Type : IPv6Address Classification: Static Accessibility: Read/write	Can be used to look up the default route in the route table. Packets that include a destination address with no corresponding entry in the routing table shall be forwarded using the default route. Field devices may use the default route to send packets to devices operating on the backbone that have not been assigned 16-bit addresses, but only IPv6Addresses
Contract_Table	8	Includes information to construct the network header for a particular PDU flow	Type: Array of NLContractTbl structures (see NLContractTbl) Classification: Static Accessibility: Read/write Valid range: See Table 207	The contract table consists of the destination address, priority for a particular flow. The table also includes information indicating if a ContractID needs to be carried in the NPDU or not. The scope of a contract ID is local to a device, however, the combination of a source address (128-bit) and ContractID is globally unique. The contract table structure and its corresponding alerts and methods are discussed below
Address_Translation_Table	9	Includes the IPv6Address and the DL16Address alias for the IPv6Addresses	Type: Array of NLATTbl structures (see NLATTbl) Classification: Static Accessibility: Read/write Valid range: See Table 209	The address translation table in a device is indexed via the IPv6Address and stores the corresponding DL16Address alias within the D-subnet to which the device belongs
Max_NSDU_size	10	Maximum service data unit size supported by the NL of the device	Type: Unsigned16 Classification: Constant Accessibility: Read only Default value: 70 Valid range: 70..1 280	Fixed value based on device memory capabilities and implementation details. The value 1 280 comes from IETF RFC 4944. See 6.2.8 on how this value can necessitate fragmentation at the AL. NSDUs that exceed the maximum value may be rejected by the device

Table 206 *(3 of 3)*

Standard object type name: NL management object (NLMO()				
Standard object type identifier: 123				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute type**	**Description of behavior of attribute**
Frag_Reassembly_Timeout	11	Amount of time (in seconds) a reassembly buffer needs to be held open for fragments to come in before discarding the NPDU	Type: Unsigned16 Classification: Static Accessibility: Read/write Default value: 60 Valid range: 1..600	The default value is 60 s, but the system manager can change this value
Frag_Datagram_Tag	12	Current tag number for fragmentation at the device	Type: Unsigned16 Classification: Dynamic Accessibility: Read only Default value: Uniform random	A new tag number is used for every NPDU that needs to be fragmented. The Tag number is incremented by one for each NPDU to be fragmented. Value wraps back to zero after 65 535
NLRouteTblMeta	13	Metadata for Route Table attribute (attribute 5)	Type: Metadata_attribute Classification: Static Accessibility: Read only	Metadata containing a count of the number of entries in the table and capacity (the total number of rows allowed) for this table. The size of the routing table present in devices that only operate within the D-subnet and are not backbone capable is presented in Table B.18
NLContractTblMeta	14	Metadata for Contract Table attribute (attribute 8)	Type: Metadata_attribute Classification: Static Accessibility: Read only	Metadata containing a count of the number of entries in the table and capacity (the total number of rows allowed) for this table
NLATTblMeta	15	Metadata for Address Translation Table attribute (attribute 9)	Type: Metadata_attribute Classification: Static Accessibility: Read only	Metadata containing a count of the number of entries in the table and capacity (the total number of rows allowed) for this table
DroppedNPDUAlertDescriptor	16	Describes how a dropped NPDU alert is reported on the network	Type : Alert report descriptor Classification: Static Accessibility: Read/write Default value: [TRUE, 7]	—

[a] The attribute shall remain unchanged during normal operation; only a system manager may change it. Assignment of a new DL16Address shall trigger a Join_Command = 3, restartAsProvisioned, following which the device shall re-join the wireless network.

10.4.2 Structured management information bases

The NLMO defines three structured management information bases (SMIBs) as tables. They are the contract table, the route table (RT), and the address translation table (ATT).

Table 207 specifies the elements for the contract table. Devices that are not backbone capable may elide the Source_Address field.

Table 207 – Contract table structure

Standard data type name: NLContractTbl		
Standard data type code: 441		
Element name	**Element identifier**	**Element scalar type**
Contract_ID*	1	Type: Unsigned16 Classification: Static Accessibility: Read/write
Source_Address*	2	Type: IPv6Address Classification: Static Accessibility: Read/write Default value : 0
Destination_Address	3	Type: IPv6Address Classification: Static Accessibility: Read/write Default value : 0
Contract_Priority	4	Type: Unsigned2 Classification: Static Accessibility: Read/write Default value : 00
Include_Contract_Flag	5	Type: Boolean1 Classification: Static Accessibility: Read/write Default value : FALSE
NOTE * indicates an index field.		

Table 208 specifies the elements for the route table.

Table 208 – Route table elements

Standard data type name: NLRouteTbl		
Standard data type code: 442		
Element name	**Element identifier**	**Element scalar type**
Destination_Address*	1	Type: IPv6Address Classification: Static Accessibility: Read/write
Next_Hop	2	Type: IPv6Address Classification: Static Accessibility: Read/write
NWK_HopLimit	3	Type: Unsigned8 Classification: Static Accessibility: Read/write Default value : 64
Outgoing_Interface	4	Type: Unsigned1 Classification: Static Accessibility: Read/write Default value: 0 Named values: 0: DL; 1: Backbone
NOTE * indicates an index field.		

Table 209 specifies the elements for the address translation table.

Table 209 – Address translation table structure

Standard data type name: NLATTbl		
Standard data type code: 443		
Element name	**Element identifier**	**Element scalar type**
Long_Address*	1	Type: IPv6Address Classification: Static Accessibility: Read/write
Short_Address	2	Type: DL16Address Classification: Static Accessibility: Read/write
NOTE * indicates an index field.		

10.4.3 NL management object methods

Standard methods such as read and write can be used for scalar or structured MIBs in their entirety. These methods are used to manipulate tables. They allow access to a particular row of a structured MIB based on a unique index field. All devices shall be capable of manipulating the NLMO attributes immediately. Devices support delayed manipulation of these attributes using cutover methods but are not required to do so.

It is assumed that the tables have a unique index field, which may either be a single element or the concatenation of multiple elements. The index field is assumed to be the

(concatenation of the) first (few) element(s) of the table. For example, the contract table index field is the concatenation of the Contract_ID and Source_Address.

These methods do not specify the interface between the NME and NLMO, as this interface is internal to a particular device. The management entity may keep local copies of the MIBs.

Table 210 describes the methods for manipulation of structured MIBs. These methods are based on the Read_Row, Write_Row, and Delete_Row templates defined in Annex J.

Table 210 – NLMO structured MIB manipulation methods

Standard object type name: NL management object (NLMO)		
Standard object type identifier: 123		
Method name	**Method ID**	**Method description**
Set_row_RT	1	Method to set (either add or edit) the value of a single row of the route table. The method uses the Write_Row method template defined in Annex J with the following arguments: Attribute_ID :5 (route table) Index 1: 1 (Destination_address)
Get_row_RT	2	Method to get the value of a single row of the route table. The method uses the Read_Row method template defined in Annex J with the following arguments: Attribute_ID :5 (route table) Index 1: 1 (Destination_address)
Delete_row_RT	3	Method to delete a single row of the contract table. The method uses the Delete_Row method template defined in Annex J with the following arguments: Attribute_ID :5 (route table) Index 1: 1 (Destination_address)
Set_row_ContractTable	4	Method to set (either write or edit) the value of a single row of the contract table. The method uses the Write_Row method template defined in Annex J with the following arguments: Attribute_ID :8 (contract table) Index 1: 1 (ContractID) Index 2: 2 (Source Address)
Get_row_ContractTable	5	Method to get the value of a single row of the contract table. The method uses the Read_Row method template defined in Annex J with the following arguments: Attribute_ID :8 (contract table) Index 1: 1 (ContractID) Index 2: 2 (Source Address)
Delete_row_ContractTable	6	Method to delete the value of a single row of the contract table. The method uses the Delete_Row method template defined in Annex J with the following arguments: Attribute_ID :8 (contract table) Index 1: 1 (ContractID) Index 2: 2 (Source Address)
Set_row_ATT	7	Method to set (either add or edit) the value of a single row of the Address Translation table. The method uses the Write_Row method template defined in Annex J with the following arguments: Attribute_ID :9 (AT table) Index 1: 1 (Long Address)
Get_row_ATT	8	Method to get the value of a single row of the Address Translation table. The method uses the Read_Row method template defined in Annex J with the following arguments: Attribute_ID :9 (AT table) Index 1: 1 (Long Address)
Delete_row_ATT	9	Method to delete a single row of the Address Translation table. The method uses the Delete_Row method template defined in Annex J with the following arguments: Attribute_ID :9 (AT table) Index 1: 1 (Long Address)

Table 211 describes the alert to indicate a dropped PDU/PDU error.

Table 211 – Alert to indicate dropped PDU/PDU error

Standard object type name(s): NL managegment object (NLMO)					
Standard object type identifier: 123					
Description of the alert: Alert to indicate dropped PDU /PDU error					
Alert class (Enumerated: alarm or event)	**Alert category (Enumerated: device diagnostic, comm. diagnostic, security, or process)**	**Alert type (Enumerated: based on alert category)**	**Alert priority (Enumerated: high, med, low, journal only)**	**Value data type**	**Description of value included with alert**
0 = Event	1 = Comm. diagnostic	0 = NL_Dropped_PDU	7 =Medium	Type: OctetString	The value is an octet string consisting of at least two octets. The first octet is an Unsigned8 that specifies the size of the value field included with the alert in octets. The second octet is an Unsigned8 that conveys the diagnostic class. Named values: 0: reserved; 1: destination unreachable; 2: fragmentation error; 3: reassembly timeout; 4: hop limit reached; 5: header errors; 6: no route, next hop unreachable; 7: out of memory; 8: NPDU size too large; 9..255: reserved. The remaining octets include the NPDU header for the dropped PDU

10.5 NPDU formats

10.5.1 General

Each NPDU shall consist of two basic components:

- A network header possibly comprising addressing, class of service (CoS), and fragmentation fields.
- A network payload of variable size containing the data that needs to be transmitted.

This standard shall allow three different NPDU header formats:

– the basic header;

– the contract-enabled header; and

– the full header.

Devices compliant with this standard shall use dispatch prefixes (the least significant 3 bits of the first octet of the NPDU) to distinguish between these header formats (see Figure 106). The prefix 000 shall indicate that the NPDU header format is the basic header. The basic header prefix conforms to the NALP dispatch of 6LoWPAN.

The prefix 011 shall indicate the contract-enabled header. The contract-enabled header conforms to the LOWPAN_IPHC dispatch of 6LoWPAN.

The prefix 010 shall indicate the full header format. Resource constrained devices that operate in the wireless D-subnet and are backbone capable may construct the full header but are not required to do so. If a packet that is constructed using the full header is received by a device that is not backbone capable, the device may discard the NPDU and send a dropped PDU/PDU error alert with value 5 indicating a header error.

Finally, a prefix of 110 or 111 shall indicate that the PDU is a fragment of a larger NPDU that needs to be reassembled.

This standard primarily uses 16-bit addresses for very first (originator) and final destination in the DPDUs transmitted over the IEEE 802.15.4 wireless network. Therefore, the network header is recommended to be either the basic or contract-enabled header format for the NPDUs transmitted over the IEEE 802.15.4 wireless links. It is not recommended, although not prohibited, to use the full header format for transmission of NPDUs over the field medium.

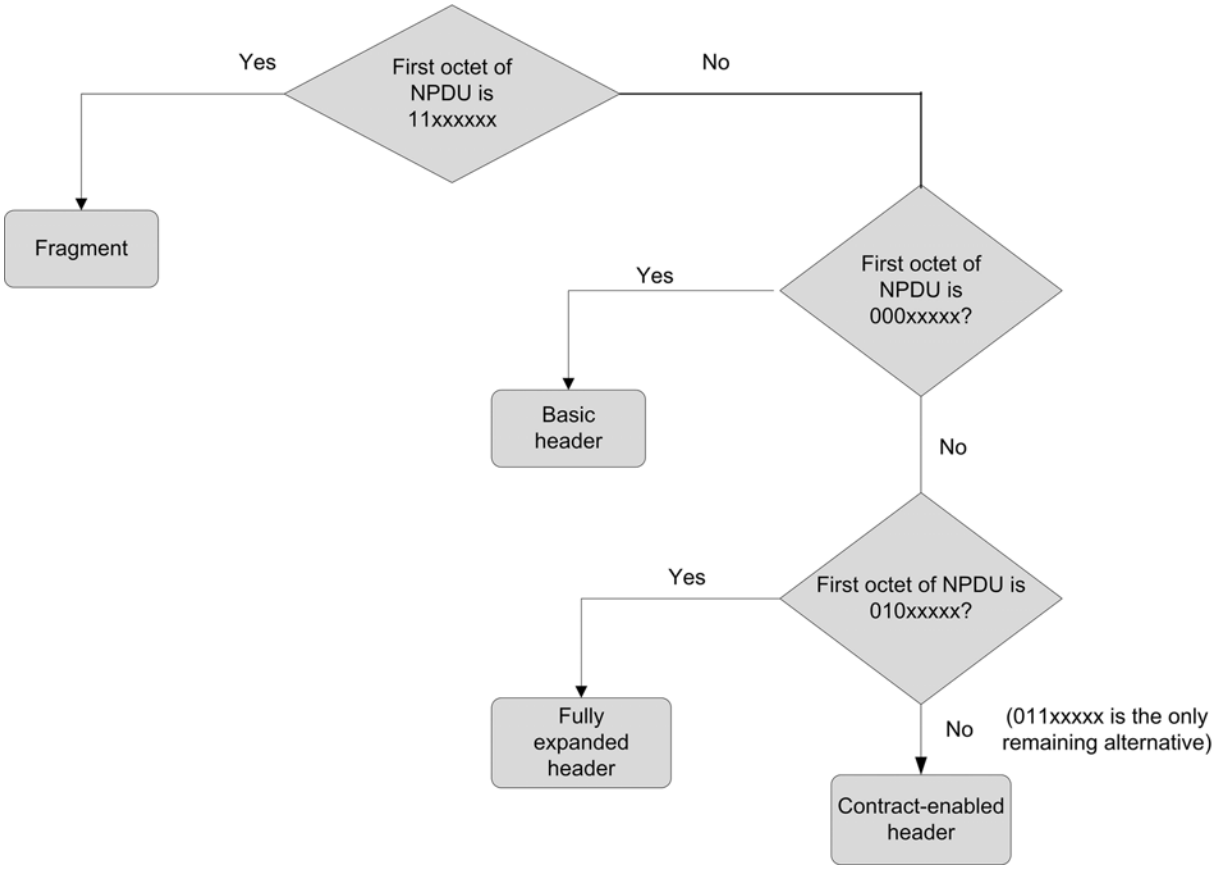


Figure 106 – Distinguishing between NPDU header formats

The dispatch octet bit patterns are shown in Table 212.

Table 212 – Common header patterns

Dispatch pattern	Header type
000xxxxx	NALP (Not A 6LoWPAN NPDU) – Basic header
010xxxxx	IPv6 (uncompressed IPv6 header) – Full header
011xxxxx	LoWPAN_IPHC (compressed IPv6 header) – Contract-enabled header
11xxxxxx	LoWPAN fragment

The NL headers shall follow the formats defined herein.

10.5.2 Basic header format for NL

10.5.2.1 Intended usage

The DL of this standard employs a link level mesh. In the most common case, a PDU will traverse a single D-subnet, so the basic header is optimized to minimize the NPDU overhead. The route that needs to be taken by the PDU is known to the device of ingress into the D-subnet; this device of ingress makes all the necessary DL routing decisions. The ContractID is not transmitted in the basic network header.

The basic header for the NL shall be used only if the user datagram protocol (UDP) header is fully compressed (i.e., the source and destination port numbers are compressed to four (4) bits each and the UDP checksum is elided). The NL can determine whether the UDP header is fully compressed by looking at the LOWPAN_NHC octet, which is always the first octet of the NSDU passed to the NL by the TL. Since the basic header is used only in case of fully compressed UDP header (i.e., fixed and known value of UDP LOWPAN_NHC) the UDP LOWPAN_NHC octet shall be elided by the NL of origin and restored by the destination NL.

10.5.2.2 Format

Table 213 describes the basic network header format.

Table 213 – Basic NL header format

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Dispatch							
(variable)	Network payload							

The basic NL header shall consist of a single Dispatch field as follows:

- Dispatch: The dispatch field indicates the NL header format. For the basic header, the dispatch field shall have the value 0x 0000 0001.

10.5.2.3 Relation to 6LoWPAN

The 6LoWPAN format allows all fields of the full IPv6 header to be elided. The dispatch and encoding octets to achieve this are 0x0111 1110 0111 0111, which indicate (when parsed as 011.11.1.10.0.1.11.0.1.11):

- Dispatch = 011
- TF = 11 (both traffic class and flow label elided)
- NH = 1 (next header field elided)
- HLIM = 10 (HopLimit = 64)

- CID = 0 (no context identifier extension)
- SAC = 1 (stateful source address compression; the ATT provides the context)
- SAM = 11 (source address fully elided)
- M = 0 (no multicast)
- DAC = 1 (stateful destination address compression; the ATT provides the context)
- DAM = 11 (destination address fully elided)

The 6LoWPAN format also allows the UDP header to be compressed so that the source and destination port numbers are four (4) bits each and the checksum is elided. In this case, the UDP LOWPAN_NHC field has the value 0x 1111 0111, which indicates (when parsed as 11110.1.11):

– protocol = 11 110 (UDP)
– checksum compression = 1 (checksum elided)
– port compression = 11 (source and destination ports are compressed to four (4) bits each and their implied prefix is 0xF0B)

The basic header is essentially a single-octet abbreviation for the three octets (two octets of fully compressed IP header and one octet of fully compressed UDP header) noted above. Since it is an abbreviation, it is fully compatible with the 6LoWPAN format. A device receiving a basic header NPDU can expand the basic header dispatch octet to the three octets noted above and obtain a 6LoWPAN-compliant PDU.

10.5.3 Contract-enabled network header format

10.5.3.1 Intended usage

Like the basic header, the contract-enabled network header is intended for use within D-subnets. The very first (originator) device of an NPDU may use the contract-enabled header instead of the basic header if it is desirable for devices other than the very first (originator) device to be aware of the NPDU stream to which the NPDU belongs. For example, an intermediate router on the backbone may need to select:

- appropriate backbone resources upon egress from the originating D-subnet; or
- appropriate DL resources (graph ID, priority, etc.) upon ingress into a destination D-subnet.

The contract includes a flag to indicate to the originating NL whether the network header requires inclusion of the ContractID.

The contract-enabled header shall also be used if the UDP LOWPAN_NHC does not indicate full compression of the UDP header. Joining-process messages between the new device and the advertising router will always fall under this category since they do not elide the UDP checksum.

10.5.3.2 Format

Table 214 describes the contract-enabled header format.

Table 214 – Contract-enabled NL header format

Number of octets	Bits							
	7	6	5	4	3	2	1	0
2	LOWPAN_IPHC dispatch			LOWPAN_IPHC encoding (bits 8..12)				
	LOWPAN_IPHC encoding (bits 0..7)							
0 or 3	Reserved				FlowLabel (bits16..19)			
	Flow Label (bits 8..15)							
	Flow Label (bits 0..7)							
0..1	HopLimit							
(variable)	Network payload							

Fields include:

- LOWPAN_IPHC dispatch: This field shall indicate that the header format is contract-enabled and that LOWPAN_IPHC header compression encoding bits follow. The LOWPAN_IPHC dispatch field shall be 011.
- LOWPAN_IPHC encoding: This field is 13 bits long; its value shall be encoded as 0x0 11HH 0111 0111 when octets 3..5 are present to carry the contract ID, or as 0x1 11HH 0111 0111 when octets 3..5 are elided. In either case, HH shall have the value 00 if HopLimit is carried inline, 01 if the hop limit is 1, 10 if the hop limit is 64 and 11 if the hop limit is 255.
- FlowLabel: The lower order 16 bits of the FlowLabel shall be set to ContractID. The higher order 4 bits shall be all zeros. This field shall only be present if octets 3 through 5 are present, as indicated by LOWPAN_IPHC encoding.
- HopLimit: This field shall indicate the number of layer-3 hops permitted before the NPDU is discarded. The HopLimit field shall be set to a value indicated by the device's routing table (RT; see 10.2.6.2). The default value for the HopLimit field shall be 64. Devices that only operate within the D-subnet and are not backbone capable shall set the HopLimit to 1. From the perspective of the NL, any device reachable through the DL mesh is a single network hop away.

For joining-process messages, LOWPAN_IPHC encoding shall have the value 0x1 1101 0111 0111 to indicate that octets 3 through 5 are elided (no contract ID) and that the hop limit is 1 (HopLimit field elided).

10.5.3.3 Relation to 6LoWPAN

Table 215 shows the 6LoWPAN_IPHC encoding format.

Table 215 – 6LoWPAN_IPHC encoding format

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	0	1	1	TF		NH	HLIM	
1	CID	SAC	SAM		M	DAC	DAM	
…	Non-compressed fields							

The encoding for the contract-enabled NL header is derived by using the following values for the 6LoWPAN_IPHC encoding fields:

- TF = 01 or 11. In a 6LoWPAN format, TF = 01 implies a 3-octet inline field is carried in the non-compressed fields. This inline field consists of 2 bits of ECN, followed by a 2-

bit pad, followed by 20 bits of flow label. Since in this standard the ECN is always enabled and the congestion indication is carried by the lower layer, both the ECN and the 2-bit pad shall be all zeros. In a 6LoWPAN format, TF = 11 implies this 3-octet field is elided.

- NextHeaderNH = 1 to indicate that the next header can be inferred from the prefix of the transport header. In this standard, the next header is always UDP.
- HLIM = HH. These bits are used by this standard to indicate the hop limit compression scheme as intended in 6LoWPAN.
- CID = 0 to indicate no additional 8-bit context identifier extension is used.
- SAC = 1 to indicate stateful compression for the source address.
- SAM = 11 to indicate that all 128 bits of the source address are elided (since, in this standard, the 16-bit D-alias is carried by the lower layer and is indexed in the ATT, which provides the translation context).
- M = 0 to indicate the destination address is not multicast.
- DAC = 1 to indicate stateful compression for the destination address.
- DAM = 11 to indicate that all 128 bits of the source address are elided (since, in this standard, the 16-bit D-alias is carried by the lower layer and is indexed in the ATT, which provides the translation context).

10.5.4 Full header (IPv6) format

10.5.4.1 Intended usage

The full header format is used primarily over the backbone network. A field device may also use the full header format instead of the basic or contract-enabled format but it is not recommended.

When the NPDU reaches an intermediate backbone router over the DL, the backbone router shall fully expand the header to include all fields as defined in the IPv6 header. It is necessary to expand the NL addresses of the very first (originator) device and the final destination device to 128 bits to disambiguate their DL16Addresses when routing outside the D-subnet.

10.5.4.2 Format

Table 216 describes the IPv6 header format.

Table 216 – IPv6 NL header format

Number of octets	bits							
	7	6	5	4	3	2	1	0
1	Version				TrafficClass (bits 7..4)			
3	TrafficClass (bits 3..0)				FlowLabel (bits 19..16)			
	FlowLabel (bits 15..8)							
	FlowLabel (bits 7..0)							
2	PayloadSize (bits 15..8)							
	PayloadSize (bits 7..0)							
1	NextHeader							
1	HopLimit							
16	Source address							
16	Destination address							
(variable)	Network payload							

Fields include:

- Version: 4-bit IP version number shall be set to 6.
- TrafficClass: The higher order four bits of this 8-bit field shall be used to carry the 4-bit priority of the NPDU over the backbone. The fifth bit shall be 0 and the sixth bit shall be used to carry the Discard Eligible (DE) bit of the NPDU over the backbone. The two lowest-order bits shall carry the ECN.
- FlowLabel: The value of this field shall be as defined in the contract-enabled header format (see 10.5.3.2). This field shall be set to all zeros if the Contract ID is not carried as part of the flow.
- PayloadSize: This 16-bit unsigned integer shall contain the size of the IPv6 payload, i.e., the rest of the NPDU following this header, in octets.
- NextHeader: This 8-bit field shall bet set to 0x 0001 0001 (17 decimal) identifying the following header as UDP.
- HopLimit: The value of this 8-bit field shall be set to the number of NL (layer 3) hops needed to get to the destination. If the NPDU is received over a D-subnet with the basic header, this field shall be updated upon egress from the D-subnet by a backbone router according to the routing table of the router. This field is decremented by 1 by the NL of each device if the NL forwards the NPDU. The NPDU shall be discarded if HopLimit is decremented to zero.
- Source address: This field shall contain the IPv6Address of the originator of the NPDU.
- Destination address: This field shall contain the IPv6Address of the final destination (intended recipient) of the NPDU.

10.5.4.3 Relation to 6LoWPAN

It is not recommended to use the full NL header format in an NPDU being transmitted over the D-subnet of this standard. However, the standard does not preclude the use of the full header over D-subnets. If used over the D-subnet, the NPDU shall contain the IPv6 dispatch as shown in Table 217.

Table 217 – Full NL header in the DL

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	IPv6 dispatch							
4	Version				TrafficClass (bits 7..4)			
	TrafficClass (bits 3..0)				FlowLabel (bits 19..16)			
	FlowLabel (bits 15..8)							
	FlowLabel (bits 7..0)							
2	PayloadSize (bits 15..8)							
	PayloadSize (bits 7..0)							
1	NextHeader							
1	HopLimit							
16	Source address							
16	Destination address							
(variable)	Network payload							

10.5.5 Fragmentation header format

10.5.5.1 Intended usage

If an NPDU with size greater than the dlmo.maxDSDUSize needs to be transmitted over the DL, the NPDU shall be fragmented. When an NPDU needs to be fragmented, the fragmentation header shall be inserted.

10.5.5.2 Format

The fragmentation header contains a 5-bit fragmentation type, followed by a 27-bit header for the first fragment and a 35-bit header for subsequent fragments. Fields of the basic, contract-enabled, or full headers from the dispatch octet onwards shall be placed in the first fragment only. Table 218 shows the NL header format for fragmented NPDUs.

Table 218 – NL header format for fragmented NPDUs

Number of octets	Bits							
	7	6	5	4	3	2	1	0
4..5	Fragmentation type					Fragmentation header		
	Other fields of fragmentation header (see Table 219 and Table 220)							
(variable)	Basic/contract-enabled/full header (for first fragment only)							
(variable)	Network payload							

Fields include:

- Fragmentation type: This 5-bit field shall be set to 0x1 1000 for the first fragment and to 0x1 1100 for the second and subsequent fragments.
- Fragmentation header:
  – First fragment: The first fragment shall contain the first fragment header as defined in Table 219 and the following text.

Table 219 – Format of first fragment header

Number of octets	Bits							
	7	6	5	4	3	2	1	0
2	Fragmentation type					Datagram_size (bits 10..8)		
	1	1	0	0	0			
	Datagram_size (bits 7..0)							
2	Datagram_tag							

  – Subsequent fragments: The second and subsequent fragments (up to and including the last) shall contain a fragmentation header as defined in Table 220 and the following text.

Table 220 – Format of second and subsequent fragment headers

<table>
<tr><th rowspan="2">Number of octets</th><th colspan="8">Bits</th></tr>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td rowspan="3">2</td><td colspan="5">Fragmentation type</td><td colspan="3" rowspan="2">Datagram_size (bits 10..8)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td colspan="8">Datagram_size (bits 7..0)</td></tr>
<tr><td>2</td><td colspan="8">Datagram_tag</td></tr>
<tr><td>1</td><td colspan="8">Datagram_offset</td></tr>
</table>

All fragment headers contain:

– Datagram_size: This 11-bit field encodes the size of the entire NSDU before fragmentation plus the size of the basic, contract-enabled or full header. The value of Datagram_size shall be the same for all fragments of the NSDU.

– Datagram_tag: This provides the identification for the reassembling NLE. The originating NLE shall increment Datagram_tag for successive, fragmented NPDUs. The incremented value of Datagram_tag is left-truncated to 16 bits (i.e., modulo 2^{16}) for inclusion in the NPDU. To minimize its predictability to an attacker, the initial value for Datagram_tag shall be selected from a uniform-random distribution. The value of Datagram_tag shall be the same for all fragments of an NSDU.

Second and subsequent fragment headers also contain:

– Datagram_offset: This field shall be present only in the second and subsequent fragments and shall specify the offset, in units of 8 octets, of the fragment from the beginning of the NPDU payload. (The first fragment of the NSDU has an offset of zero; the implicit value of Datagram_offset in the first fragment is zero.) This field is 8 bits long.

10.5.5.3 Relation to 6LoWPAN

The fragmentation header formats in this standard are based entirely on the IETF RFC 4944 format, with no changes. As in IETF RFC 4944, the Datagram_size field is carried in all fragments, and the Datagram_offset is expressed in 8-octet units. Fragmentation and reassembly can occur at intermediate devices and are not necessarily end-to-end.

11 Transport layer

11.1 General

The reference model in this standard is composed of five protocol layers. In this model, transport is the fourth layer, immediately subordinate to the AL. A TLE responds to service requests from a superior ALE at a TSAP and issues service requests to an inferior NLE at an NSAP.

The TL is responsible for end-to-end communication and operates only at the communication endpoints (as opposed to the routing devices).

The TL provides connectionless services, usually with cryptographic-based security:

- The connectionless service extends UDP over IPv6, as defined in IETF RFC 768 and IETF RFC 2460, by providing an alternative, more secure TPDU integrity check with cryptographic authentication and, when so configured, privacy.
- Security is handled in a manner similar to that of the DL, but from end-to-end rather than hop-by-hop.

- The connectionless service also extends 6LoWPAN as defined by IETF RFC 6282. When security is activated, it is possible to compress the UDP header to one octet by eliding the UDP checksum and relying on the larger, appended transport message integrity code (TMIC) to provide end-to-end integrity.

11.2 TLE reference model

The TLE reference model is shown in Figure 107.

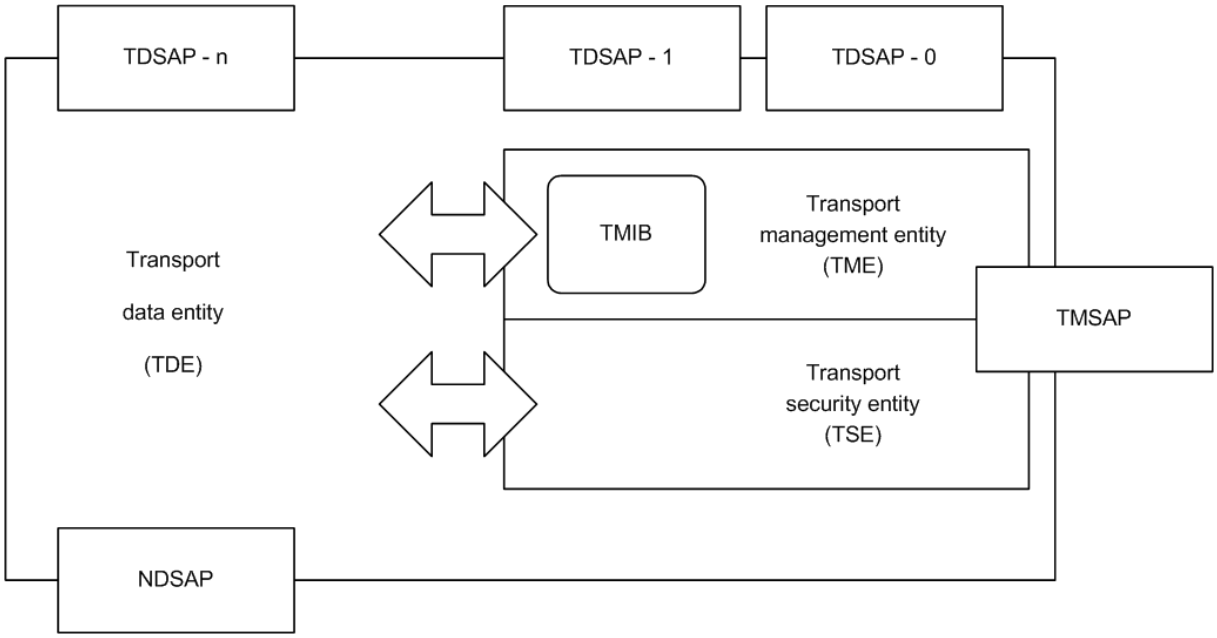


Figure 107 – TLE reference model

The TLE conceptually includes the transport management entity (TME), the transport data entity (TDE) and the transport security component (TSC).

The TDE has a dedicated TDSAP for the DMAP, TDE-0; a TDSAP that is reserved for SMAP, TDE-1; and a TDSAP for each UAP, TDE-2 to TDE-n.

The TME configures and monitors the actions of the TLE. The transport management information base (TMIB) (see 11.6.2.5.2) maintains abstract information for the TME, that is accessed at the TMSAP as part of the TLMO. The TSC provides the TLE's security functions, based on tables of security information that are maintained and monitored via the TMSAP. The TDE uses the TSC to perform security operations on TPDUs.

11.3 Transport security entity

11.3.1 General

The TSC is conceptually a compartmented service within the TLE. TL security can protect the integrity of the conveyed transport service data unit (TSDU), the transport header and the transport endpoint IPv6Addresses. When active, it also provides protection against excessive TPDU delay and TPDU replay, and can encrypt the TSDU for confidentiality.

11.3.2 Securing the TL

The TSC and DSC share functionality. The TSC is responsible for:

a) determining which security level shall be applied to a given secure session based on policies;

b) on receipt of a TPDU (as part or all of an NSDU),
   1) discarding non-conforming TPDUs;
   2) discarding TPDUs that fail, depending on T-association configuration, either
      i) the traditional UDP integrity check that protects against non-malicious errors, or
      ii) cryptographic authentication checks that protect against both accidental and deliberate TPDU modification, excessive TPDU delay and TPDU replay;
   3) decrypting a protected TSDU conveyed by the TPDU;
c) when preparing a TPDU for transmission via a NLE,
   1) encrypting TSDUs that are to have confidentiality protection during TL conveyance; and
   2) depending on the session configuration, either
      i) including the traditional, computed UDP integrity checksum to protect against non-malicious errors, or
      ii) including cryptographic authentication material to protect against both accidental and deliberate TPDU modification, excessive TPDU delay and TPDU replay.

The functionality of the TSC is defined in 7.3.3.

Similar to DL security, TL security uses a cryptographic block cipher in a generic authenticated encryption block cipher mode called the Counter with CBC-MAC (CCM*), as defined by IEEE 802.15.4:2011, Annex B. The default block cipher is AES-128, but other block ciphers can be used where required by national regulation.

CCM* enables authentication of a message while encrypting only a part of that message, leaving the rest of the message (usually a header) in the clear. When this feature is enabled for a particular session, the UDP checksum is not used and is elided in the compressed form of the TPDU, as specified in 11.4.3.4. It is always present in the expanded form of the TPDU, to provide compliance with IETF RFC 2460 (UDP over IPv6).

11.4 Transport data entity

11.4.1 General

The TDE provides connectionless services based on the User Datagram Protocol (UDP, IETF RFC 768) over IPv6 (IETF RFC 2460), with a compression as defined in IETF RFC 6282.

The main steps in TPDU construction are, in order:

a) A TSDU is received from a local ALE through a TDSAP.

   NOTE This TSDU can convey a single APDU or multiple concatenated APDUs.

b) The TSDU is protected and timestamped as described in 7.3.3.2.1. The resulting UDP payload comprises a TL security header, the TSDU, and, when cryptographic security is configured, a TMIC. The TSDU is encrypted for confidentiality if so indicated by the TL security header. The presence of the TMIC, and its size, are conveyed to receiving device(s) in the TL security header.
c) A UDP header is prepended to the UDP payload. The UDP header may be compressed. Compression involves eliding the UDP checksum from the UDP header when the UDP payload contains an alternative integrity check. An integrity check within the UDP payload may be a TMIC as indicated in the TL security header and/or an integrity check embedded in a tunneled payload.

11.4.2 UDP over IPv6

UDP (IETF RFC 768) provides a connectionless mode of computer communication in the environment of an interconnected set of computer networks.

UDP is essentially transparent to application operation, leaving both control and responsibility for proper network operation to the AL. Proven, standard-based approaches exist for this purpose. Relevant issues are further documented in 11.4.4.

IETF RFC 768 assumes that IPv4 is used as the underlying protocol and defines the computation of a UDP checksum, used for error detection, that covers a UDP pseudo-header of information from the IPv4 network header, the UDP header, and the UDP payload. If that computation yields a result of zero, it is changed to 0xFFFF for placement in the UDP header and the value of zero is reserved to indicate that there is no checksum computation.

IETF RFC 2460 specifies the changes to IETF RFC 768 to adapt UDP for IPv6. The changes to UDP are minor and relate to the computation of the UDP checksum, which becomes mandatory and includes a modified UDP pseudo-header adapted for IPv6, as shown in Figure 108. Per IETF RFC 2460, IPv6 receivers discard UDP packets containing a checksum of 0x0000 and log this event as an error.

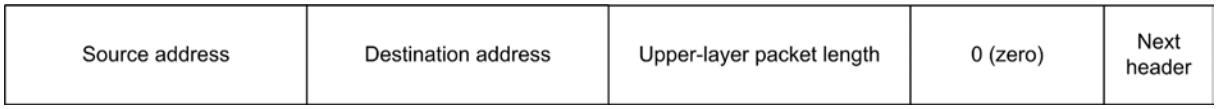


Figure 108 – UDP pseudo-header for IPv6

In the pseudo-header, the value in the field named ‘Next header’ is set to 17 to indicate UDP and the value in the field named ‘Upper-layer packet length’ is the same as the Length field in the UDP header and accounts for the size of the whole TPDU, including the UDP header.

A different pseudo-header, the TSS pseudo-header is used in TL security processing, as described in 7.3.3.2.1.

NOTE See IETF RFC 768, IETF RFC 2460, and 4.5.2.1 and 7.3.3.2.1 to understand the purposes of these pseudo-headers.

11.4.3 UDP header transmission and compression

11.4.3.1 General

The TL supports uncompressed UDP for both transmission and reception of a TPDU. A device that implements the TL of this standard shall support the uncompressed UDP, but should compress the UDP header for transmission over the wireless network.

Table 221 describes the UDP header encoding.

Table 221 – UDP header encoding

Number of octets	Bits (presented in IEC convention, which is different from IETF convention)						
	7	6	5	4	3	2	1..0
2	Source port: that maps one to one with the source device TDAP						
2	Destination port: that maps one to one with the destination device TDAP						
2	Size: the size in octets of this user datagram, including this header and the APDU						
2	Checksum: the 16-bit one's complement of the one's-complement sum of a) the UDP pseudo-header of Figure 108, derived from the IP header; b) the UDP header; and c) the possibly-encrypted TSDU, padded with zero octets at the end (if necessary) to make a multiple of two octets. The checksum field of the UDP TPDU is first set to zero during the checksum computation.						

The TL also complies with 6LoWPAN, which specifies a method for compressing UDP using a Next Header Compression (LoWPAN_NHC) format and mandates the use of UDP compression over the local D-subnet. A device that implements the TL of this standard shall thus support all the combinations offered by the LOWPAN_NHC for compression and expansion of the UDP header.

The maximum size of a TSDU depends on a variety of factors, such as the buffering capacity at each session endpoint TLE, the selected TL security level, and network characteristics. The system manager configures TSDU maximum size on a per-contract basis to account for these and any other relevant considerations, through the value of Assigned_Max_TSDU_Size, as described in Table 30.

11.4.3.2 Compressing and restoring UDP port numbers

Compression for UDP port numbers is described in IETF RFC 6282:2011, 4.3. Those port numbers that are optimized by the compression process should be assigned to the ports (AEs) with highest frequency of use. Optimum compression is obtained when both the source and destination port numbers start at a base number P and are expressed as P + short_port, where:

- P is the base port, $61\,616_{10}$, that is, 0xF0B0;
- short_port is a positive integer that is ≤ 15 (i.e., 0x0F).

In such a case, the pair of source and destination ports is compressed to a single octet that specifies the source and destination values as short_ports, which reduces the size of the TPDU field required to convey the UDP port numbers from four octets to one octet.

When it is not possible to have both ports within the 0xF0B0..0xF0BF range, it is still possible to reduce the TPDU size by one octet if either the source or the destination port is in the 0xF000..0xF0FF range; in that case the corresponding high-order octet of 0xF0 is elided. It is thus recommended that server applications listen to a port in the 0xF000..0xF0FF range. For example, a typical field device has a DMAP at 0xF0B0 (encoded as short_port 0) and may have its single UAP at 0xF0B2 (encoded as short_port 2), since 0xF0B1 (encoded as short_port 1) is reserved for the local SMAP.

11.4.3.3 Eliding and restoring the UDP Size field

If the UDP header is not compressed, then the NSDU size is equal to the TPDU size that is placed in the UDP header.

If the UDP header is compressed, the UDP Size field is always elided in transmission, and is inferred upon reception from the NSDU size passed by the receiving NLE. In that case, the

TPDU size is equal to the NSDU size plus the computable difference between the sizes of a full UDP header and the actual UDP header as compressed.

Even if none of the other fields in the UDP header are compressed or elided, compressing the UDP Size field reduces the TPDU size by one octet and enables the use of the basic header at the NL.

For example, in the optimal case described in Table 223, the compressed UDP header requires 2 octets, thus saving 6 octets compared with the fully-expanded UDP header. The TDPU size for the expanded UDP header is thus NSDU_Size + 6.

11.4.3.4 Eliding and restoring the UDP checksum

The TL complies with the 6LoWPAN rules and operations defined in IETF RFC 6282:2011, 4.3.2, as follows:

- The authorization to elide the UDP checksum might come from the ALE or the TSC. The TSC hints at the presence or absence of this checksum by specifying whether the TMIC is present or not.
- An ALE may elide the UDP checksum in the following cases:
  - Tunneling: the source ALE is tunneling a PDU (of unspecified type) that possesses its own integrity mechanism that provides at least as much protection as the 16-bit UDP checksum.
  - Application MIC: the source ALE applies an end-to-end message integrity check of at least 16 bits (e.g., as part of a key exchange).
- If the local NLE is a backbone router, then the router shall recompute the elided checksum based on the received packet as specified in IETF RFC 2460 and shall place the result of that computation in the reformed UDP header before transmission over an IPv6 backbone network.
- If the NLE is the destination of the packet and is aware of the presence of the TMIC in the TPDU, then it may omit the UDP checksum operation completely (neither recompute nor check the checksum).
- A backbone router that forwards an IPv6 packet into D-subnet uses the security control field in the TPDU security header to determine whether a TMIC is present or not. When performing the UDP compression, the backbone router should elide the UDP checksum if the TMIC is present but it shall not elide the UDP checksum if the TMIC is not present. Conversely, the fact that the checksum is not compressed is not an indication that there is no TMIC in the TPDU.

6LoWPAN permits UDP checksum compression only when the alternative protection within the TPDU provides integrity protection at least as great as the UDP checksum, including end-to-end protection for all the fields that the UDP checksum would protect. To meet that requirement, this standard defines a TSC pseudo-header that is included in the TMIC computation, as described in 7.3.3.2.1. Compared to the standard UDP pseudo-header of IETF RFC 2460, the TSC pseudo-header includes the UDP ports but does not include the payload size, since that is implicitly protected by the TMIC. The structure of the TSC pseudo-header is defined in Table 47.

In the case of a TPDU being prepared for transmission, the source and destination IPv6Addresses are obtained from the contract information; the contract ID itself is passed by the ALE as a parameter associated with the TSDU. The Next header field of the pseudo-header is set to 17 (UDP). The source port is obtained from the TL context associated with the TSAP, whereas the destination port is another parameter associated with the TSDU. The TSC pseudo-header has been modeled as if it were being constructed by the TDE and passed together with the TSDU to the TSC for security processing.

The TSC uses the TSC pseudo-header in the TMIC computation by prepending it to the TSDU, but the TSC pseudo-header is not encrypted when encryption is to be performed on

the TSDU. Once the cryptographic process is completed, the TSC shall remove the TSS pseudo-header from the processed TSDU, add its own headers and trailers, and return the protected UDP payload to the TDE. The TDE shall complete the TPDU by prepending, and usually compressing, the UDP header to form the NSDU that is passed to the NL.

In the case of a received TPDU, the source and destination IPv6Addresses and the priority bits are passed by the NL as parameters associated with the NSDU, and the ports are obtained from the UDP header in the TPDU once the UDP header is expanded. The TDE shall fill the pseudo-header, remove the UDP header from the NSDU, and pass the resulting protected UDP payload together with the pseudo-header and the priority bits to the TSC for security processing.

Since an NPDU might be received out of sequence, perhaps due to its priority settings (see Table 205), the TSC may use that priority information to partition its look-aside cache, potentially optimizing its ability, within limited memory resources, to validate TPDUs that incurred substantial transit delay. The priority information is conveyed with the NSDU in the N-DATA.indication received by the TLE from a local NLE; the TLE forwards that information and the TPDU to the TSC.

The receiving TSC strips the TL security headers and trailers from the protected TSDU, prepends the TSC pseudo-header, UDP ports, and security headers (see Figure 41), and then performs the security verifications. If an integrity check is applied, the TLE can verify whether any of the critical parts of the NL and TL information was modified during end-to-end transport.

If the UDP upper-layer checksum field is elided, it is up to the receiving device on the wireless D-subnet to recompute the UDP checksum as part of the process of reversing the 6LoWPAN compression, if necessary, before further forwarding of the conveying NPDU to a destination outside the wireless D-subnet. This compression reversal step is needed in particular by a middlebox, such as a BBR border router, that forwards the uncompressed NPDU onto a different network.

NOTE It is useless for the addressed destination TLE to reconstitute the uncompressed TPDU.

If the UDP checksum is not elided, then the UDP checksum shall be computed in transmission after the security operation is complete and the security fields are preset for the computation. The UDP payload that is used for the UDP checksum shall include all data from the UDP header to the end of the TPDU, including the security fields, application data, and MIC fields, whether the application data is encrypted or not. Upon reception, it shall be verified by the TDE prior to TSC operation unless it is known by the TDE that the TSC verifies the MIC, in which case the checksum may be ignored.

11.4.4 TSAPs and UDP ports

A device autoconfigures one link-local IPv6Address based upon its EUI64Address. After the subnet joining process is complete, a device also has at least one global IPv6Address. This standard permits, but does not require, a device to support more than one global IPv6Address. A UDP port shall be associated with no more than one UAP for a given IPv6Address.

The port is used as the local port for all transmissions from/to that UAP using that IPv6Address. Further multiplexing between UAP objects is the responsibility of the ALE, done within TSDUs, and thus is transparent to the TLE. As a result, only a limited number of ports are actually used in the system, and there is no dynamic port allocation after the (re-)initialization phase of the applications.

Each UAP shall be responsible for knowing its UDP port number and registering it with the TLE. At that time, a TLSAP shall be mapped on a one-to-one basis with the UAP and the local port for the given IPv6Address. An application process address shall consist of a device address concatenated with its TLSAP identifier.

11.4.5 Good network citizenship

(N)-SDUs are exchanged with the upper and lower layers through (N)-DATA primitives. An implementation can report transient congestion in a lower layer via a specific completion code in an (N)-DATA.confirm primitive that is conveyed all the way up the protocol suite. Upon receiving a completion code indicating transient congestion, an ALE should either delay retransmission of the (N)-SDU by an exponential backoff amount or simply drop the (N)-SDU.

Since the TL is based on UDP, it is desirable that the UAPs support TCP-Friendly Flow Control or TCP-Friendly Rate Control (TFRC, IETF RFC 5348), depending on the nature of the flow, in order to protect future uses of the network and to share the backbone network fairly.

This is usually achieved by implementing a blocking protocol at the AL that enforces a round-trip time (RTT) between TPDUs or uses RTT to compute a loss recovery timer. In the case of periodic samples, this may be achieved either by reserving bandwidth or by keeping the sampling period above the initial RTT value for TCP of 3 s (see IETF RFC 6298). Higher sample rates may be used in some applications.

11.5 TPDU encoding

11.5.1 General

A TPDU exchanged between the two TLEs of a communication shall be composed of a UDP header in its uncompressed form, a security information header, the TSDU, and a TMIC, as shown in Figure 109.

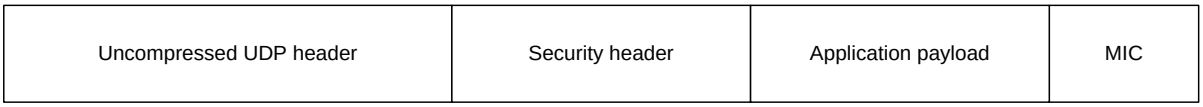


Figure 109 – TPDU structure

The NSDU data passed by the sending TLE to a local NLE via an NSAP includes any UDP header compression that has been applied to the TPDU. If compression was applied, the 6LoWPAN_NHC-for-UDP octet is the first octet of the NSDU.

The pair of IPv6Addresses, the transport type (UDP=17) and the size of the NSDU are not present in the NSDU, so are passed as associated parameters. Whether 6LoWPAN_NHC compression took place is also passed as a parameter.

11.5.2 Header compression – User datagram protocol encoding

When the UDP header is not compressed, the NSDU is identical to the TPDU and the UDP header shall be present in full without a prefixed 6LoWPAN_NHC-for-UDP encoding octet.

If any of the UDP header fields is compressed, a 6LoWPAN_NHC-for-UDP encoding octet shall be prepended that describes the compression. The NSDU shall be formed by replacing the UDP header at the beginning of the TPDU with the 6LoWPAN_NHC-for-UDP encoding octet followed by the compressed data. The 6LoWPAN_NHC-for-UDP encoding octet is structured as in Table 222.

Table 222 – UDP 6LoWPAN_NHC-for-UDP encoding octet

Number of octets	Bits (presented in IEC convention, which is different from IETF convention)							
	7	6	5	4	3	2	1	0
1	1	1	1	1	0	C	P	

The C (checksum compression) and P (ports compression) fields are defined as follows:

C field (1 bit):

0: The TPDU's 16-bit checksum is conveyed in the TPDU.

1: The TPDU's 16-bit checksum is elided from the TPDU.

P field (2 bits):

00: The TPDU conveys the source and destination ports as 16-bit values.

01: The TPDU conveys the source port as a 16-bit value while only the low-order 8 bits of the destination port, which is in the range 0xF000..0xF0FF, are conveyed in the TPDU.

10: The TPDU conveys the destination port as a 16-bit value while only the low-order 8 bits of the source port, which is in the range 0xF000..0xF0FF, are conveyed in the TPDU.

11: The TPDU conveys only the low-order 4 bits of the source and destination ports, which are in the range 0xF0B0..0xF0BF.

Compressed fields appear in the same order as they do in the UDP header format specified in IETF RFC 768.

When the highest degree of compression is achieved, only the compressed short_port numbers are carried after the 6LoWPAN_NHC-for-UDP encoding octet as shown in Table 223.

Table 223 – Optimal UDP header encoding

Number of octets	Bits (presented in IEC convention, which is different from IETF convention)						
	7	6	5	4	3	2	1..0
1	1	1	1	1	0	1 (checksum is elided)	11 (source and destination ports are compressed)
2	source short_port				destination short_port		

The expanded values for the compressed source port and destination ports shall be calculated using the formula:

UDP_port_number = P + short_port

where short_port is the 4-bit compressed port number and P is the value 0xF0B0 ($61\,616_{10}$). In other words, short_port conveys the low-order 4 bits of the UDP port number.

At the time of the subnet joining process, the field device does not have the relevant cryptographic material to compute a TMIC. Because this situation requires that the TMIC be omitted, the device shall compute the UDP checksum as prescribed by IETF RFC 768 and IETF RFC 2460 and shall use the link-local IPv6Addresses to compute and send the checksum and transmit. In this case, if the UDP ports can be compressed, the encoded UDP header is formatted as represented in Table 224.

Table 224 – UDP header encoding with checksum and compressed port numbers

Number of octets	Bits						
	7	6	5	4	3	2	1..0
1	1	1	1	1	0	0 (checksum is present)	11 (source and destination ports are compressed)
2	source short_port				destination short_port		
3	UDP checksum						
4							

11.5.3 TPDU security header

For information on the TPDU security header, encoding, and decoding, see 7.3.3.6.

11.6 TL model

11.6.1 General

A TLE provides two interfaces:

- A TDE TDSAP for each UAP and one for the DMAP.
- A TME TMSAP for the DMAP.

These interfaces are illustrated in the protocol reference model of this standard, shown in Figure 16.

All interfaces between the TLE and adjacent layer entities or management entities are internal interfaces within the device, and thus are unobservable. Therefore they are not subject to standardization.

Upper layers use the TDSAP to exchange data communicated via the TLE. There is a separate TDSAP instance for each UAP, plus an instance for the DMAP.

The TDSAP includes a multiplexer function that adapts the address namespace of the ALE to the native address namespace of the TLE. The DMAP uses the TMSAP to configure the TLE and monitor its operation.

The TLE communicates with a local NLE using an NDSAP.

11.6.2 Data services

11.6.2.1 General

For illustration purposes, an example set of primitives is provided in this standard. The TL offers an unconnected service based on the UDP model.

A TLE uses the interface supplied by the TDSAP to transmit and receive protocol data units with the ALE.

The TDSAP transfers the TSDU, along with control and status information parameters.

11.6.2.2 T-DATA.request

11.6.2.2.1 General

T-DATA.request instructs the TL to transmit a protocol data unit.

11.6.2.2.2 Semantics of the service primitive

The semantics of T-DATA.request are as follows:

```
T-DATA.request   (
                 ContractId
                 TSDU_size
                 TSDU
                 Priority
                 DE
                 TDSAP
                 \Destination_Port)
```

Table 225 specifies the elements for T-DATA.request.

Table 225 – T-DATA.request elements

Element name	Element identifier	Element scalar type
ContractId (identifies the contracted TL resources associated with the protocol data unit)	1	Unsigned16 Named values: 0: no contract
TSDU_size (size in octets of the protocol data unit passed from the ASL)	2	Unsigned16
TSDU (the TSDU to be transmitted)	3	OctetString
N-priority (identifies the priority within the NL of this TSDU)	4	Unsigned2
DE (identifies the Discard Eligibility of this TSDU)	5	Unsigned1
TDSAP (ID of the TDSAP that grants UDP service over SourcePort)	6	Unsigned16
Destination_Port (UDP destination (remote) port for the TSDU)	7	Unsigned16

The TLE maintains a table indexed by TDSAP that contains (implicitly or explicitly) the local IPv6Address and (explicitly) the local port for transmission. That table is accessed to obtain the source IPv6Address and port for transmission over a given TDSAP. The destination IPv6Address is obtained from the Destination_Address field in the contract information, indexed by the contract ID.

The TLE does not retain state information related to the remote port; thus, information that is passed by the UAP is placed in the TPDU without checking.

The N-priority parameter communicates the N-priority that is to be used by the NL. The N-priority settings are not used in the TL, but they are passed on through the NL to the DL, where they are used to select the priority class in the DL header.

11.6.2.2.3 Appropriate usage

An ALE invokes the T-DATA.request primitive to pass a TSDU to a local TLE for transmission on the network.

11.6.2.2.4 Effect on receipt

On receipt of the T-DATA.request primitive, the TLE looks up the T-security level in the KeyDescriptor to determine the processing required by the local TSC, constructs the TPDU header, forms the TPDU, and generates the N-DATA.request to a local NLE at an NSAP.

11.6.2.3 T-DATA.confirm

11.6.2.3.1 General

T-DATA.confirm is used by the TLE to respond to a T-DATA.request. The confirmation is immediate and tells the requesting ALE either that the request was successful or that an error was detected. Error conditions include such issues as an unrecognized contract ID, a TSDU size that is incorrect or excessive, or an internal transient error such as network congestion beyond the capacity of the TSC and its agents to cope.

11.6.2.3.2 Semantics of the service primitive

The semantics of T-DATA.confirm are as follows:

```
T-DATA.confirm    (
                        ContractId
                        TDSAP
                        status
                        )
```

Table 226 specifies the elements for T-DATA.confirm.

Table 226 – T-DATA.confirm elements

Element name	Element identifier	Element scalar type
ContractId (identifies the contracted TL resources associated with the TPDU)	1	Type: Unsigned16 Named values: 0: no contract
TDSAP (ID of the TDSAP that grants UDP service over SourcePort)	2	Type: Unsigned16
Status (the result of the T-DATA.request primitive)	3	Type: Unsigned Valid range: (see Table 227)

Table 227 provides the status codes for T-DATA.confirm.

Table 227 – T-DATA.confirm status codes

Name	Description
SUCCESS	TSDU accepted
TRANSIENT_FAILURE	TSDU rejected, but can be retried after a short period of time
FAILURE	Generic failure: TSDU rejected without explicit reason
INVALID_CONTRACT	Specific failure: Unrecognized contract ID; TSDU rejected
INVALID_LENGTH	Specific failure: TSDU is larger than Assigned_Max_TSDU_Size; rejected
PORT_ERROR	Specific failure: SourcePort is not registered for TDSAP; TSDU rejected
SAP_ERROR	Specific failure: Unknown TDSAP; TSDU rejected

11.6.2.3.3 When generated

A TLE generates T-DATA.confirm in response to a local T-DATA.request. The T-DATA.confirm returns synchronously either a status of success or the appropriate error code.

11.6.2.3.4 Appropriate usage

The T-DATA.confirm notifies the ALE of the result of its request to transmit a TSDU.

11.6.2.4 T-DATA.indication

11.6.2.4.1 General

T-DATA.indication is used to transfer a TSDU to the ALE. It is generated when a TPDU has been successfully received from a local NLE and processed by the TLE.

A received TPDU that fails T-security processing or that specifies an unregistered destination port is discarded on receipt. Such errors are logged in the TLE's PIB.

11.6.2.4.2 Semantics of the service primitive

The semantics of T-DATA.indication are as follows:

```
T-DATA.indication (
                    SrcAddr
                    SrcPort
                    TSDU_size
                    TSDU
                    ECN
                    TDSAP
                    transportTime
                    )
```

Table 228 specifies the elements for T-DATA.indication.

Table 228 – T-DATA.indication elements

Element name	Element identifier	Element scalar type
SourceNetworkAddress (IP address of the remote end)	1	Type: IPv6Address
SourcePort (UDP source port in incoming TPDU)	2	Type: Unsigned16
TSDU_size (size in octets of the accompanying TSDU)	3	Type: Unsigned16
TSDU (the received higher-layer content of the TPDU)	4	Type: OctetStringN
ECN (explicit congestion notification bits)	5	Type: Unsigned2
TDSAP (ID of the TDSAP that grants UDP service and matches a local port)	6	Type: Unsigned8
TransportTime (end-to-end transit time from originating to receiving TSC)	7	Type:Unsigned16
NOTE TransportTime is related to tpduMaxAge.		

A TLE maintains a table that contains the TDSAP, which is indexed by (implicitly or explicitly) the local address and (explicitly) the local port for reception. That table is accessed to find the TDSAP used to pass the TSDU to the UAP.

11.6.2.4.3 Appropriate usage

A TLE invokes T-DATA.indication to notify the addressed ALE of a received TSDU.

11.6.2.4.4 Effect on receipt

On receipt of T-DATA.indication, the ALE processes the received TSDU.

11.6.2.5 Management services

11.6.2.5.1 General

The TLE's management service is controlled by the TL management object (TLMO) in the DMAP. The TLMO controls the TLE functionalities by:

- measuring TL latency and making the related adaptations to comply with latency requirements dynamically; and
- collecting operational parameters.

The TLMO uses its TMSAP interface to configure and control the operation of the TLE. The TLE's TME provides a TMIB that maintains the state information necessary to implement the TMSAP functionality.

11.6.2.5.2 Attributes

Table 229 specifies the attributes of the TLMO.

Table 229 – TLMO attributes *(1 of 2)*

Standard object type name: TL management object (TLMO)				
Standard object type identifier: 122				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Reserved	1	Reserved by the standard	—	—
MaxNbOfPorts	2	Number of active ports	Type: Unsigned8 Classification: Constant Accessibility: Read only Default value: Device-dependent Valid range: 2..255	The minimum value covers a typical device with a single DMAP and a single UAP TDSAP
TPDUin	3	Counter reporting the number of received TPDUs	Type: Unsigned32 Classification: Dynamic Accessibility: Read only Default value: 0	Incremented with each TPDU received from a remote TLE
TPDUinRejected	4	Counter reporting the number of rejected TPDUs	Type: Unsigned32 Classification: Dynamic Accessibility: Read only Default value: 0	Incremented with each data unit received from a remote TLE that was discarded (e.g., for security reasons). Note that there is no such counter within the DSMO

Table 229 *(2 of 2)*

Standard object type name: TL management object (TLMO)				
Standard object type identifier: 122				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
TSDUout	5	Counter reporting the number of TSDUs passed to a local ALE	Type: Unsigned32 Classification: Dynamic Accessibility: Read only Default value: 0	Incremented with each TPDU received from a remote TLE that resulted in the conveyance of a contained TSDU to a local ALE
TSDUin	6	Counter reporting the number of received TSDUs	Type: Unsigned32 Classification: Dynamic Accessibility: Read only Default value: 0	Incremented with each TSDU received from a local ALE
TSDUinRejected	7	Counter reporting the number of rejected TSDUs	Type: Unsigned32 Classification: Dynamic Accessibility: Read only Default value: 0	Incremented with each TSDU received from a local ALE that is rejected
TPDUout	8	Counter reporting the number of TPDU passed to the NL	Type: Unsigned32 Classification: Dynamic Accessibility: Read only Default value: 0	Incremented with each TSDU received from a local ALE that is conveyed to and accepted by a local NLE
IllegalUseOfPortAlertDescriptor	9	Used to change the priority of IllegalUseOfPort alert; this alert can also be turned on or turned off	Type: Alert report descriptor Classification: Static Accessibility: Read/write Default value: [TRUE, 8] -- medium	—
TPDUonUnregisteredPortAlertDescriptor	10	Used to change the priority of TPDUonUnregisteredPort alert; this alert can also be turned on or turned off	Type: Alert report descriptor Classification: Static Accessibility: Read/write Default value: [TRUE, 4] -- low	—

For each attribute, the TLMO provides read and write methods available to the DMAP. Those methods are implemented by requesting TME services across the TMSAP.

11.6.2.5.3 Methods

In addition to the read and write service for the attributes, additional TLMO methods provide access to TME services across the TMSAP.

Table 230 describes the reset method.

Table 230 – TL management object methods – Reset

Standard object type name: TL management object (TLMO)				
Standard object type identifier: 122				
Method name	**Method ID**	**Method description**		
Reset	1	Reset the transport to initial states		
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Forced	Boolean	Forced means that all data are updated without any interaction with other entities. TSAP-related tables are emptied, related memory is freed and all counters are set to 0
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Status	Unsigned8	0 = success, > 0 failure

Table 231 describes the halt method.

Table 231 – TL management object methods – Halt

Standard object type name: TL management object (TLMO)				
Standard object type identifier: 122				
Method name	**Method ID**	**Method description**		
Halt	2	Halts a port and places it back in its initial state. Similar to a reset, but scoped to one UDP port only. All traffic is interrupted on that port and the TSAP needs to be reopened for that port to become operational again. The TSAP-related table entries for the port are emptied and related memory is freed		
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	DeviceAddress	IPv6Address	This device IPv6Address
	2	PortNumber	Unsigned16	The port to halt
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Status	Unsigned8	Named values: 0: success, 1: generic failure, 2: bad port number

Table 232 describes the PortRangeInfo method.

Table 232 – TL management object methods – PortRangeInfo

Standard object type name: TL management object (TLMO)				
Standard object type identifier: 122				
Method name	**Method ID**	**Method description**		
PortRangeInfo	3	Reports the UDP ports range information		
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	DeviceAddress	IPv6Address	This device IPv6Address
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Status	Unsigned8	0 = success, > 0 failure
	2	NbActivePorts	Unsigned16	Number of active ports
	3	FirstActivePort	Unsigned16	First active port
	4	LastActivePort	Unsigned16	Last active port

Table 233 describes the GetPortInfo method.

Table 233 – TL management object methods – GetPortInfo

Standard object type name: TL management object (TLMO)				
Standard object type identifier: 122				
Method name	**Method ID**	**Method description**		
GetPortInfo	4	Reports the UDP port information for a given UDP port or the first active UDP port		
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	DeviceAddress	IPv6Address	This device IPv6Address
	2	PortNumber	Unsigned16	The port whose info is requested (0 = lowest)
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Status	Unsigned8	0 = success, > 0 failure
	2	PortNumber	Unsigned16	This port number
	3	UID	Unsigned32	Owner application ID
	4	Compressed	Boolean	Compression applies to this port
	5	TPDUsInOK	Unsigned32	Number of TPDUs accepted over the port
	6	TPDUsInKO	Unsigned32	Number of TPDUs rejected over the port
	7	TPDUsOutOK	Unsigned32	Number of TPDUs sent over the port
	8	TPDUsOutKO	Unsigned32	Number of TPDU transmission failures

Table 234 describes the GetNextPortInfo method.

Table 234 – TL management object methods – GetNextPortInfo

Standard object type name: TL management object (TLMO)				
Standard object type identifier: 122				
Method name	**Method ID**	**Method description**		
GetNextPortInfo	5	Reports the UDP port information for a given UDP port or the first active UDP port		
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	DeviceAddress	IPv6Address	This device IPv6Address
	2	PortNumber	Unsigned16	The previous port from which info is requested
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Status	Unsigned8	0 = success, > 0 failure
	2	PortNumber	Unsigned16	The port for which info is reported
	3	UID	Unsigned32	Owner application ID
	4	Compressed	Boolean	Whether compression applies to this port
	5	TPDUsInOK	Unsigned32	Number of TPDUs accepted over the port
	6	TPDUsInKO	Unsigned32	Number of TPDUs rejected over the port
	7	TPDUsOutOK	Unsigned32	Number of TPDUs sent over the port
	8	TPDUsOutKO	Unsigned32	Number of TPDU transmission failures

11.6.2.5.4 Alerts

Table 235 describes the alert to indicate illegal use of a port by an application.

Table 235 – TL management object alert types – Illegal use of port

Standard object type name(s): TL management object (TLMO)					
Standard object type identifier: 122					
Description of the Alert: Alert to indicate illegal use of a port by an application					
Alert class (Enumerated: alarm or event)	**Alert category (Enumerated: device diagnostic, comm. diagnostic, security, or process)**	**Alert type (Enumerated: based on alert category)**	**Alert priority (Enumerated: high, med, low, journal only)**	**Value data type**	**Description of value included with alert**
0 = Event	1= Communication diagnostic	0 =IllegalUseOfPort	8 = Medium	Type: Unsigned16	16-bit port number

Table 236 describes the alert to notify of a received TPDU that addresses an unregistered port.

Table 236 – TL management object alert types – TPDU received on unregistered port

Standard object type name(s): TL management object (TLMO)					
Standard object type identifier: 122					
Description of the Alert: Alert to notify of a TPDU received on unregistered port					
Alert class (Enumerated: alarm or event)	**Alert category (Enumerated: device diagnostic, comm. diagnostic, security, or process)**	**Alert type (Enumerated: based on alert category)**	**Alert priority (Enumerated: high, med, low, journal only)**	**Value data type**	**Description of value included with alert**
0 = Event	1 = Communication diagnostic	1=TPDUonUnregisteredPort	4 = Low	Type: OctetString	First 40 octets of the TPDU

Table 237 describes the alert to notify of a received TPDU that does not meet local security policies.

Table 237 – TL management object alert types – TPDU does not match security policies

Standard object type name(s): TL management object (TLMO)					
Standard object type identifier:					
Description of the Alert: Alert to notify of a TPDU that does not match security policies					
Alert class (Enumerated: alarm or event)	**Alert category (Enumerated: device diagnostic, comm. diagnostic, security, or process)**	**Alert type (Enumerated: based on alert category)**	**Alert priority (Enumerated: high, med, low, journal only)**	**Value data type**	**Description of value included with alert**
0 = Event	1 = Communication diagnostic	2=TPDUoutOfSecurityPolicies	2 = Journal	Type: OctetString	First 40 octets of the TPDU

12 Application layer

12.1 General

The application layer (AL) defines software objects to model real-world objects, and also defines the communication services necessary to enable object-to-object communication between distributed applications in an open, interoperable application environment compliant with, and based on, this standard. This standard does not define the operation of the distributed applications themselves; that is, neither the local operation of the application itself, such as the manner in which an application acquires the values of the object attributes it supports for access, nor direction regarding how and when an application applies the models and/or the services defined herein, are addressed by this standard.

NOTE 1 For example, a real-world analog input is modeled as an AnalogInput object. The AnalogInput object often communicates its process variable to a correspondent party by using the AL-provided publish service.

The AL supports wireless devices in the field, as well as gateways that integrate a wireless network compliant with this standard and its devices with a host control system.[8]

8 See 5.2.6.5 for a more complete discussion of the roles supported by this standard.

The application model in this standard is specifically designed to satisfy the constraints of wireless communication environments.

NOTE 2 An object oriented AL approach is used for the following key reasons.

Command-based protocols are able to be designed to conform to the object model defined by this standard by describing the commands as separate object methods. That is, a command-based application is able to be modeled using the object model in this standard.

The object model supports well accepted architectural principles of logical information separation. For example, management information is logically separate from operating data. Operational information for independent variables is logically separate. In order to maintain separation of information, the protocol is required to identify the corresponding object. This adds a one-octet overhead to identify the object, which was deemed to be a more than reasonable approach for architectural separation of information.

12.2 Energy considerations

The need to extend battery life makes energy-efficient messaging extremely important. The use of battery power or energy scavenging/harvesting techniques for connected field devices requires additional considerations in communication layer design, compared to the approach taken for wired devices. Not only does every communicating layer need to consider device resource availability, but it also needs to consider energy consumption minimization (within architecturally appropriate constraints of course) in order to extend battery life or to operate within the scavenging/harvesting budget.

Since energy is consumed by message processing, as well as by the basic control operation of the device, it is necessary to balance communications efficiency with employing proven and well-accepted architectural principles of information separation as well as message processing efficiency. The native application model in this standard is defined to meet these needs.

12.3 Legacy control system considerations

Wireless networks compliant with this standard may be connected to legacy control systems.

NOTE 1 A model that integrates with existing systems makes possible the reuse of existing and proven tools and interfaces, and also reduces overall development and test time, resulting in earlier production of robust implementations.

An application process in a native device communicates over the network using only ASL-defined services and payloads. An application process in a non-native device requires communication of constructs that are not defined by the ASL service payloads. This communication from the non-native device over the network defined by this standard is accomplished by using the subset of standard services defined in this standard that support communication of payloads that are not explicitly and entirely defined by this standard.

The native AL defines special objects and services to support non-native protocol tunneling to meet system integration needs.

The set of defined application objects that support non-native defined payloads are the tunnel object and the interface object. The set of standard-defined application services that support payloads beyond those defined within this standard are the tunnel service, which is used for aperiodic communication, and the publish service using non-native mode, which is used for periodic communications. For example, the payload of the tunnel service for aperiodic messaging in order to encapsulate data constructed by a legacy system is not defined within WISN.

NOTE 2 One way to achieve a synthesis with existing systems is to create an energy- and resource-optimized version of a wire-oriented existing legacy approach by mapping the legacy model to the model in this standard and directly employing the native AL. Such mappings usually are defined by individual protocol consortia, such as the HART Communication Foundation (HCF), Fieldbus Foundation (FF), PROFIBUS Nutzerorganisation (PNO), ODVA,

which supports managed protocols such as CIP (Common Industrial Protocol), and others.[9] Another way to achieve synthesis is to use a protocol tunnel through the native AL. For both approaches, energy and resource implications are important considerations.

NOTE 3 Whichever method is chosen, the higher-level system still communicates with wireless devices within a network compliant with this standard.

NOTE 4 Electronic device description files often are used to meet this requirement. For this standard, a device description language (DDL) or an electronic device description language (EDDL) describe native devices.

NOTE 5 See Annex R for further details regarding host system interface.

12.4 Overview of object-oriented modeling

12.4.1 General

An object model is a protocol-, platform-, and language-neutral means of describing and distinguishing components (system elements) that have a unique identity. Objects separate the world into meaningful and manageable pieces. Not only do object definitions promote modularity, but they also promote component reusability. An object can represent anything that has a state and a behavior; objects expose attributes to represent state, and provide methods that operate on the object's state to effect particular behaviors. For example, device-specific methods that may be supported by a DMAP may include device-specific self-test methods or device reset methods.

12.4.2 Object-to-object communication concept

From the user's point of view, AL communication occurs from one object in an application process to another object in an application process. Concepts of polymorphism allow the same communication techniques to be applied to this standard's industry-independent user application objects and industry-dependent objects, as well as to this standard's management objects.

In keeping with this principle, the application model defines both an object model and a communication interaction model (service and protocol). The application model also supports multiple application processes within a device, each of which may contain multiple standard objects. This enables this standard to meet specific market needs, such as for process industries or factory automation, as well as to enable support for both single-processor and multi-processor device architectures.

The application object may, for example use the services of the AL to:

- read the value of an attribute of a remote object;
- write the value of an attribute of a remote object;
- request execution of an object-specific method of a remote object;
- report an alert related to a remote object;
- acknowledge an alert reported by a remote object;
- publish data to a remote object by using scheduled communication bandwidth;
- tunnel a non-WISN-native application message to a remote object.

Coding for these services can be found in 12.23.1.4.

---

[9] HART, FF-H1, PROFIBUS, and ODVA are the trademarks of various trade organizations. This information is given for the convenience of users of the standard and does not constitute an endorsement of the trademark holders or any of their products. Compliance to this profile does not require use of the registered trademark. Use of the trademarks requires permission of the trade name holder.

12.4.3 AL structure

The AL is divided into two sublayers, the upper AL (UAL) and the application sublayer (ASL), as shown in Figure 16. There is a one-to-one relationship between an ASLDE-SAP and a TLDE-SAP.

The UAL contains the application processes for the device. These processes may be represented as a UAP or as a management process (MP), for example the DMAP or other logical management application such as a security management application.

UAPs may be used, for example, to:

- handle input and/or output hardware;
- distribute communications to a set of co-resident UAPs within a device (proxy function);
- support tunneling of a non-native (e.g., control system legacy) protocol compatible with the network environment of this standard; and/or
- perform a computational function.

A UAP may perform an individual function or any combination of functions. How a UAP accomplishes these functions internally is beyond the scope of this standard. The AL is concerned with application-specific message content, the externally visible behavior of the standard objects contained within the UAP, and the logical interfaces to the ASL that represent UAP communication to and from the lower communication protocol suite of this standard. UAL processes may contain one or more objects that communicate with one another over the network described in this standard, using the standard services provided by the ASL.

NOTE How the ASL implements internal message routing or inter-application communication within a device (within a UAP, across UAPs in a common processor, across UAPs in different physical processors of the same device, or between a UAP and an MP) is a local matter and hence is outside the scope of this standard.

12.4.4 UAP structure

Figure 110 depicts the overall structure of a UAP as defined by this standard.

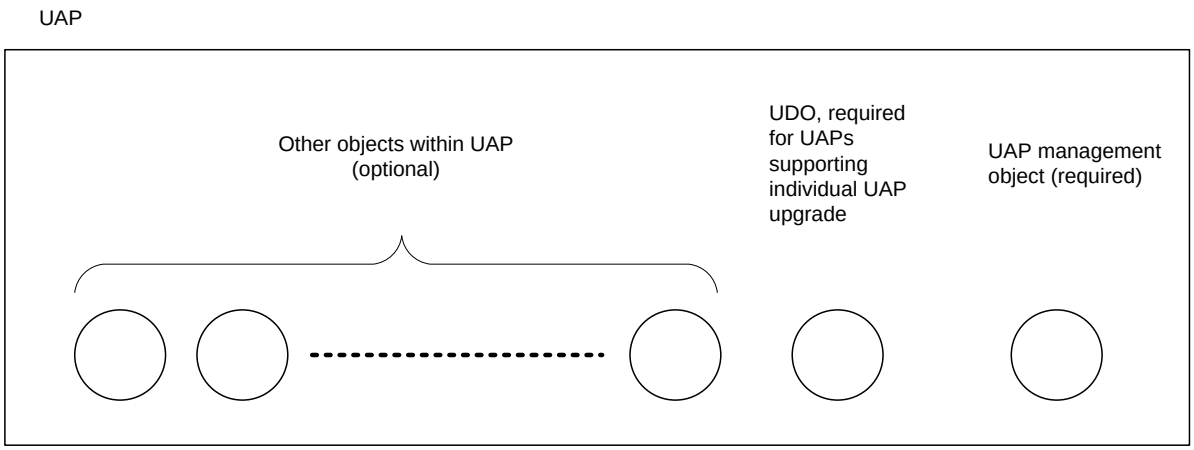


Figure 110 – User application objects in a UAP

Representation of the applications and their functions by standard object definitions allows uniform management and construction of distributed applications. The UAP management object identifies the UAP to the standard-compliant network and allows visibility of and/or control over certain operational aspects of the UAP as a whole. This UAP management object has a reserved object identifier of 1 (one). If a UAP supports individual upgrade, the UAP

shall also contain a standard UploadDownload object (UDO) to support UAP upgrade. This UDO has a reserved object identifier of 2 (two). If a UAP does not support individual UAP upgrade, the UAP shall not contain an object instance with an object identifier of 2 (two). Additional objects also may be contained within a UAP in order to provide application-specific functionality to the UAP.

NOTE 1 The UAPMO is sufficient to represent the UAP to the communication system.

NOTE 2 Other objects are statically or dynamically instantiated within the UAP.

NOTE 3 It is outside the scope of this standard to define what happens to the data contained within a UAP when the UAP is upgraded.

The interaction model describes inter-object communication, including message classification and messaging formats. The ASL contains services that support object-oriented communication and routing to the appropriate destination object within a UAP, across the network. This interaction maps the ASL to the services provided by the lower communication protocol suite layer (see Table 281 and Table 282 indicating AL use of TL services and qualities of service). Between the UAL and the ASL is an ASL data entity SAP(ASLDE-n SAP).

ASL-specific management is also locally supported via a separate management SAP.

12.5 Object model

The AL defined by this standard takes advantage of object-oriented modeling concepts to support both native protocol and non-native (legacy) protocol tunneling within applications. Non-native protocol tunneling is achieved by a specialized UAP that includes one or more tunnel objects (TUN) and protocol translation facilities. This UAP consists of exactly one UAP management object to enable uniform system and network management of the UAP, plus one or more TUNs to send/receive encapsulated messages being tunneled to the UAP. Other native objects defined by this standard may also be used within a tunneling UAP. For example, tunneling may be used for wireless device-to-wireless device communication, as well as for wireless device-to-gateway communication.

NOTE 1 Addressing constructs other than these are outside the scope of this standard. Legacy protocol APDU (as opposed to native APDU) constructs are preserved by means of encapsulation when application tunneling as defined by this standard is used.

An object-oriented approach is used to encapsulate data (attributes) and functionality (methods and internal state) for re-use and consistency. Objects are individually addressable using an object identifier that is unique within the application. This unique identifier allows the AL to route messages to the appropriate object destination. Each message is interpreted and acted on by the destination object based on the message context and content. Object operation is described in terms of the network-visible operation of the destination object that is the target of the AL service.

This standard defines standard object identifiers.

NOTE 2 The complement of standard identifiers supported by a device is indicated by the version of this standard that is supported. The supported version is available from the DMAP. See Clause 6 for further details.

An object instance that is accessible within an application process is distinguished from another instance of the same object class in the same application process by its object identifier. Different object instances may also have different attribute values and/or the conditional attributes contained in its set of supported attributes.

As an example, an application process may contain two instances of the AnalogInput object. The object instances within the application process can be distinguished by their object identifier. Each object may support different values for scaling, and also may require a different complement of alarms to be reported. Hence, the instances may have a different complement of analog descriptor attributes supported by each instance.

The objects defined in the UAPs and MPs of this standard adhere to traditional object modeling concepts; specifically, these objects contain attributes, and appropriate object-specific methods, if applicable, are defined. The AL defines standard objects to provide interoperability (within its domain of application) via access to standard attributes and invocation of standard methods.

Standard objects can be classified into usage profiles to meet the needs of particular industries.

Standard management objects are expected to be always available. Application user objects may be statically instantiated, or they may be dynamically instantiated as the result of a download operation. Standard objects are extensible in the following ways:

- Industry-specific standard object types may be added.
- Vendor-specific object types may be added.
- Industry-specific attributes may be added to standard objects.
- Vendor-specific attributes may be added to standard objects.
- Industry-specific methods may be added to objects via industry-specific profiles.
- Vendor-specific methods may be added to objects.
- Industry-specific profiles of object types may be defined, such as a process industry profile, a factory automation industry profile, and other profiles.

12.6 Object attribute model

12.6.1 General

Objects defined by this standard support two kinds of attributes, object key attributes and named attributes.

In this standard, a resource is represented as an object, and identified by an object key attribute, which is a numeric value.

NOTE 1 Support of an object key attribute using an alphanumeric representation of the resource, and of corresponding directory services to locate the resource and to resolve the external alphanumeric name form to an internal numeric form, are a subject of future standardization.

An attribute indicates an accessible element representing a property or characteristic of a resource. In this standard, an attribute is represented by a unique numerical value that uniquely identifies the attribute relative to the containing object. The supported range of the valid values for an attribute identifier is 0..4 095.

This standard defines standard object attributes. Additional standard attributes may be defined in the future.

NOTE 2 The complement of standard attributes supported is established by the version of this standard that is supported. The version of the standard that is supported is available from the DMAP.

Exposing the resource elements as attributes of an object allows the state of the object to be determined and also allows the behavior of the object to be modified. A resource attribute is defined by:

- its influence on object behavior;
- the set of values it can take (as constrained by the object definition containing the attribute);
- the valid tests (for example, valid value set matching) that may be performed on it; and
- the specific set of error conditions that may cause an object-defined failure as a result of performing an attribute-oriented operation.

Attributes themselves do not have accessible properties or subtypes. Attribute values may be explicitly established by external means or by internal means (for example, derived by computation using the values of other attributes).

Attributes shall have a data type that is a standard scalar type defined by this standard. Attributes may have a data type that is a standard data structure defined by this standard. Up to two indices are available to address constructs of standard types defined by this standard. The valid range for an index is 0..32 767.

NOTE 3 For example, access to an individual element of a singly-dimensioned array of standard scalar types is supported. As another example, access to an individual element of a data structure contained in a singly-dimensioned array of such structures is supported.

For an attribute constructed as an array, the array size shall be fixed and all elements of an array shall be homogenously sized. For example, an array of octet strings shall have the same size octet string for each element in the array. When it is necessary to indicate both the current and maximum dimension of an array, metadata information should be used. This metadata information is described by data type code 406 (Metadata_attribute data structure) which is defined in 6.2.6.3.

12.6.2 Attributes of standard objects

For each standard object, standard attributes are defined. Each standard attribute has a standard attribute identifier that is used to address the attribute.

Standard object attributes may be extended in the following ways:

- Industry-specific attributes may be added to standard objects.
- Vendor-specific attributes may be added to standard objects.

Extensions to standard attributes need to be coordinated to ensure that attribute identifiers remain unique within an object type. The mechanisms used by industries and vendors to extend the attributes of the standard object are outside the scope of this standard.

12.6.3 Attribute classification

Attributes are classified to provide guidance regarding their expected frequency of change. This information is useful, for example, to gateway devices that cache information. The frequency at which attribute values change is characterized as:

- constant;
- static;
- static-volatile;
- dynamic; or
- non-cacheable.

A constant attribute is unchanging throughout time. An example of a constant attribute is the serial number of a wireless device. Default values in this standard are all constant attributes. The values of these attributes shall be preserved when a device undergoes a warm restart/power-fail, when a device resets to factory defaults, or when a reset command to the relevant attribute or management object is received.

Constant information may be either:

- fixed information that never changes, such as information to indicate a manufacturer or serial number; and
- information that does not change during normal system operation, but that may change, for example, when a firmware download occurs.

A static attribute changes its value infrequently. Usually, static data is changed as the result of an external message, request, or event. Some static information may, for example, only be changed by a configuration tool. Operating ranges, units, communication end points, alarms, and constant input values are examples of static information. Attributes storing provisioning information, as well as configuration information provided to a device, are static attributes. The values of these attributes shall be preserved when a device undergoes a warm restart/power-fail or when a reset command to an un-related attribute or management object is received.

A static-volatile attribute changes its value infrequently. Usually, static-volatile data is changed as the result of an external message, request, or event. Some static-volatile information may, for example, only be changed by a configuration tool. The values of these attributes need not be preserved when a device undergoes a warm restart/power-fail. The values of these attributes shall be preserved when a reset command to an un-related attribute or management object is received.

A dynamic attribute may be changed spontaneously by the device containing the object and without external stimulation from the wireless network. Examples of dynamic attributes are frequently changing values such as process variables, calculations, and timers. Dynamic attributes may be treated differently by a gateway cache infrequently changing (static) values; this is entirely up to gateway internal implementation. A dynamic attribute is not required to survive when a device goes through a warm restart/power-fail or when a device resets to factory defaults. It can be reset when a reset command to the relevant attribute or management object is received.

Non-cacheable information is never buffered; for example, it may be used for critical information such as safety-related information (whose use may be outside the explicit scope of this standard), and for values that change too often to make caching a valid technique. Whenever the value of a non-cacheable attribute is requested, it shall be retrieved from the end device that owns the object and attribute.

If device local caching is needed, it is the local responsibility of the application.

12.6.4 Attribute accessibility

Network accessible attributes may be:

- accessible to be read only; or
- accessible to be both read and written.

12.7 Method model

Methods represent the set of object type-specific interfaces (functions) that can be used to access an object instance. For example, the UploadDownload object supports a StartUpload method.

The standard object methods shall always be available, and shall not be enhanced beyond the definition given by this standard.

Methods shall not be defined if the equivalent result can be achieved using a standard (object type-independent) AL service, for example, the ASL-provided read service. Definition of a method may be warranted, for example, to replace a sequence of communication transactions in order to save energy. Definition of a method may also be warranted when synchronization issues may result if individual actions are used rather than an atomic transaction set.

Standard object methods may be added in future by this standard.

Time-based triggering of application process activities is not a communication subsystem responsibility. If such time-based triggering is necessary, either a parameter of a method or a

dedicated attribute of an object may be employed. If coordination across objects is required, an application object may be defined with an attribute representing the coordinated action, acting as a proxy for the coordination.

12.8 Alert model

The term alert is used to describe an application message that advises or warns the recipient of the presence of an impending or existing situation of interest. The alert model describes alerts reported by application process resident objects and the mechanism to report them.

Two types of alerts are supported, alarms and events. Event is the term used to represent a stateless condition (that is, to indicate a situation has occurred). Events simply report that something happened. An alarm is a stateful condition of an existing situation, for example, that an alarm has transitioned to an abnormal state, or has returned to normal from an abnormal state. The alarm condition remains true until the alarm condition clears. Alert is the term used to describe the messaging of an event condition or alarm condition. Both alarms and events are reported through the alert reporting mechanism defined in this standard.

An alarm is characterized by a state, and alerts are used to report:

- the occurrence of a condition; and
- the return to normal of the previously reported condition.

Events and alarms supported by this standard fall into one of the following categories:

– device-related;
– communication-related;
– security-related; or
– control process-related.

Each alarm and event defined for an object shall have an associated attribute that describes how it is reported. This associated attribute shall include:

- whether it is enabled or disabled for reporting; and
- its priority (importance).

For all alarms, descriptive information shall also include, if the alert is an alarm, whether or not the condition is in or out of alarm.

An analog alarm occurs when a value meets or exceeds an established limit. For analog value alarms, descriptive information shall also include limit information, if any, relating to when the alarm condition is triggered.

12.9 Alarm state model

Table 238 and Figure 111 represent the alarm state model.

Table 238 – State table for alarm transitions

Transition	Current state	Event(s)	Action(s)	Next state
T1	Clear	Alarm is detected	Report alarm to ARMO in DMAP	In alarm
T2	In alarm	Clear is detected	Report alarm clear to ARMO in DMAP	Clear
T3	In alarm	Recovery requested	Report alarm to ARMO in DMAP	In alarm
T4	Clear	Recovery requested	Ignore	Clear
NOTE Recovery is usually requested by a remote device.				

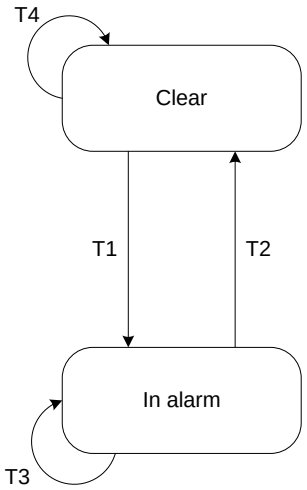


Figure 111 – Alarm state model

Alarm detection applies both to analog and discrete values. Examples of analog alarms include:

- analog limit alarms (for example, when a value exceeds a high or low threshold);
- analog deviation alarms (for example, the difference between a process variable and set point);
- a Boolean alarm (for example, when the state of the Boolean matches the discrete limit parameter); and
- diagnostics, such as those defined in the NAMUR Recommendation NE107.

NOTE 1 Alarms that depend upon evaluation of a combination of device-, inter-object-, or intra-object-specific state conditions are considered a local matter and are thus outside the scope of this standard.

NOTE 2 Different levels of alarm conditions are indicated by different alarms. For example, for an analog input, a High alarm represents one level, and a High-High alarm represents a higher level.

12.10 Event state model

12.10.1 General

The state model for an event is a subset of the state model for an alarm.

12.10.2 State table and transitions

Table 239 and Figure 112 represent the event state table and transitions, respectively.

Table 239 – State table for event transitions

Transition	Current state	Event(s)	Action(s)	Next state
T1	Clear	Event condition is detected	Determine report characteristics (e.g., priority)	Detected
T2	Detected	Event condition is reported	Report event to ARMO in DMAP	Submitted
T3	Submitted	Event condition submission to ARMO as completed	Reset to prepare for next event report	Clear

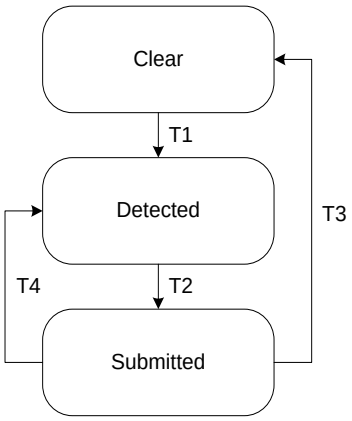


Figure 112 – Event model

12.11 Alert reporting

12.11.1 General

Alerts are reported promptly and accurately time-stamped using a queued unidirectional alert report communication. Queued unidirectional alert reporting involves the alert detecting device reporting the condition using a source/sink communication flow. A queued unidirectional alert acknowledgment is received in return.

NOTE 1 In a published message, status information sometimes indicates that an alert is available in the reporting device, accessible via a client/server read service. This method is sometimes used for factory automation; other factory automation systems publish a tag to a server that generates alarms by testing limit values in the server.

NOTE 2 Source/sink communication is used rather than client/server in anticipation of a future release of this standard that supports multicast alert reports.

12.11.2 Alert types

The alert reporting management object (ARMO) contained in the DMAP provides encapsulation of the alert report, handles timeouts and retries, and throttles alert reporting in a common manner for all the applications contained within the device.

There shall be only one ARMO in each device, in the DMAP. As described in 6.2.7.2.3, the DMAP may provide limited access to entities other than the SMAP in order to support services related to process-related alerts and device-related alerts. Alert acknowledgments shall be addressed to the ARMO.

Diagnostic alerts are specific to the device reporting them. For example, diagnostic alerts may indicate that:

- an error has occurred;
- a symptom has been detected that may indicate that an undiagnosed error occurred;
- a symptom has been detected that may indicate that an error may occur in the future;
- an error will occur if preventative action is not taken.

Diagnostic alerts may pertain to a device as a whole, to an individual component, or to a defined set of components of a device. Diagnostic alerts may be stateless or state-oriented. Diagnostic alerts may be specified by this standard, such as for communication-related alarms, or may be vendor-specific. Diagnostic alerts provide information that can later be examined to establish device and/or communication system behavior patterns.

Process alerts are specific to the process being controlled by the device reporting the process alarm. A process alarm indicates a situation in which the alarmed variable has exceeded established operational limits. For example, a process alert may be generated when a

measurable control condition occurs that is outside of desired control system operation parameters.

Process alerts provide information that can later be examined to establish control system behavior patterns, such as:

– alerts that often occur in a particular sequence;
– alerts that often occur close in time;
– alerts that were active for significant periods of time;
– actions that are required to resolve an alert situation;
– assistance in determining optimal trip point and hysteresis settings; or
– information regarding control system performance in terms of alert prevention and resolution.

Process alerts pertain to a particular control object and attribute value of that object (e.g., the PV of an analog input object). Process alerts are usually state-oriented (i.e., alarms).

One octet is used in the coding of alert type information. For all objects, three standard ranges are identified for disjoint definitions of an object-specific alert type:

00..49: reserved for and defined by this standard;

51..100: reserved for future standard industry profiles;

101..255: vendor-assignable for vendor-specific alerts.

12.11.3 Alert report information

The APDU header indicates the application and object that initiated the communication. For alert reports, this would indicate the DMAP and the ARMO. The individual alert report information therefore also shall identify the application process and the object within it that is the detecting source of the alert. Additionally, alert reports shall include the following information:

- network time of detection;
- individual alert identifier (so that duplicates may be detected by the UAL process);
- alert class (alarm or event);
- alarm direction (transition into alarm, or not (i.e., either return-to-normal or an event));
- alert category (device diagnostics, communication-related, security-related, or process-related);
- alert priority (ranges are defined for high, medium, low, and journal-only alert priorities);
- alert type (object-specific, and within the specific alert category);

NOTE See 6.2.7.2 for further information on communication alerts and 6.1.2 for further information on security alerts.

- associated-data size, in octets; and
- associated data for the alert condition.

The associated data for device diagnostics should be defined for compatibility with NAMUR Recommendation NE107; such diagnostics should indicate whether the device condition is abnormal, and if so its NAMUR class: failure, off-specification, diagnostic maintenance, or diagnostic check function.

12.11.4 Alarm state recovery

If a loss of communication with a wireless device occurs, process industries require that existing conditions be reliably recoverable by an alert receiving device, such as by determining the state when an alert receiving device starts up.

NOTE 1 It is possible for multiple alarm conditions to exist simultaneously within a process control device.

Recovery of alarm state may be requested from the ARMO. A single alarm recovery request triggers the re-reporting of all existing alarm conditions in the device of a given category. When recovering alarms, the alarm reporting device shall provide alerts to indicate when the recovery is commencing and when the recovery has completed.

NOTE 2 As event conditions are stateless, they are not recoverable.

12.12 Communication interaction model

12.12.1 General

Native communication in this standard supports both native protocol and encapsulation of legacy protocols via tunneling. The following types of communication flows are anticipated for compliant devices:

- queued unidirectional communication (e.g., alarm reporting or alarm acknowledgment);
- queued bidirectional communication (e.g., read, write, method invocation); and
- buffered unidirectional publication communication (e.g., publish).

The actual location of the buffers used to hold the data for scheduled unidirectional publication communication is a device local matter.

NOTE Buffered scheduled data publication (periodic, change of state, and application driven publication) all occur (as needed) within the scheduled phase. Communication contracts for periodic communication employ buffered unidirectional publication communication. Communication contracts for aperiodic communication employ a queued communication paradigm.

12.12.2 Buffered unidirectional publication communication

12.12.2.1 General

Buffered unidirectional communication is used when a publishing application is sending a message to a subscribing application. The buffer contains the message to be sent. On each buffered unidirectional publication contract, there is a parameter to indicate if the buffer is to always be transmitted (whether the content has been updated or not), or if the buffer is only to be transmitted if it has been updated since the prior transmission.

12.12.2.2 Buffer content always transmitted

In buffered unidirectional communication, if a publishing communication protocol suite receives another ASL publish service request for a particular communication contract before the previous message has been transmitted, the new request replaces the previous request. In the subscriber, if a new message is received before the previous one has been delivered to the application, the new message shall replace the previous undelivered message.

NOTE 1 In establishing a contract for periodic communication, the system manager ensures that there is adequate capacity within the intermediate devices along a route to support the periodic communication.

NOTE 2 It is anticipated that an application that receives an older publication after a newer one is able to choose to discard the older publication.

If a publishing communication protocol suite does not receive a new service request for the contract when it is time to transmit, the previous request shall be retransmitted. If the subscriber receives the same application service request in succession, the subscribing

application shall treat this situation as receipt of a duplicate message. Application handling of a duplicate buffered message is left to the application, and is not defined by this standard.

12.12.2.3 Buffer content transmitted on change only

This mode of buffered communication supports a change of state communication mechanism.

In buffered unidirectional communication, if a publishing communication protocol suite receives another service request for a particular communication contract before the previous message has been transmitted, the new request replaces the previous request. In the subscriber, if a new message is received before the previous one has been delivered to the application, the new message shall replace the previous undelivered message.

NOTE In establishing a contract for periodic communication, the system manager ensures that there is adequate capacity within the intermediate devices along a route to support the periodic communication.

If a publishing communication protocol suite does not receive a new service request for the contract when it is time to transmit, the previous request shall not be retransmitted. If the subscriber receives the same application service request in succession, the subscribing application shall treat this situation as an error situation. Application handling of a duplicate buffered message is left to the application, and is not defined by this standard.

12.12.3 Queued unidirectional communication

Queued unidirectional communication supports queued distribution of unconfirmed (unidirectional) ASL communication services. To satisfy this type of communication need, the lower layers of the correspondents are expected to provide a queued data transfer service.

Application handling of a duplicate AlertReport shall result in sending another AlertAcknowledgment. Receipt of a duplicate AlertAcknowledgment shall be ignored. Application handling of duplicate queued unidirectional Tunnel messages is left to the application, and is not defined by this standard.

12.12.4 Queued bidirectional communication

12.12.4.1 General

Queued bidirectional communication supports queued distribution of confirmed (bidirectional) ASL communication services. To satisfy this type of communication need, lower layers of the correspondents are expected to provide a queued data transfer service.

This maximum number of simultaneously outstanding queued bidirectional (client/server) confirmed service requests permitted for a contract is indicated to the application process by the communication contract when the contract is granted. The default value for this maximum value shall be 1, i.e., the default indicates that the contract supports only one outstanding request at any given time.

Application handling of a duplicate request is to send another response. Application handling of a duplicate response when a response is received that does not match a pending request identifier shall result in ignoring the response.

12.12.4.2 Retries and flow control

12.12.4.2.1 General

The AL defined by this standard is required to track what happens to the queued service client/server requests that it sends. This is necessary for two reasons, to ensure reliability of delivery and for flow control.

For delivery reliability, the application needs to be able to determine when it should re-send (retry) its message. There are two situations in which message retry may be necessary, first when the message request did not arrive at its final destination, and second when the message request arrived, but the application response did not make it back to the original requestor. Flow control is necessary to ensure that the destination device is not overwhelmed with messages it cannot handle, as well as to protect the network and optimize network throughput.

This standard supports both forward explicit congestion notification by the lower communication protocol suite and an application level echo back to a client of a four-part service requestor if congestion occurred on the path taken for the original service request.

To enable multiple outstanding requests simultaneously while still allowing an application to achieve both reliable delivery and flow control, each client request shall contain a unique identifier. Retransmission of a request (retries) shall use the same identifier. This identifier enables the application to implement a sliding window technique to control the flow. The client shall start a service related time-out timer when it initiates a client service request. This timer shall be based on round trip times (RTT) for messages, and shall allow sufficient time for a message from an application in device X to reach the destination application in device Y, for the server application in device Y to issue a response, and for the response to travel back to the service requesting application in device X.

NOTE This method is commonly known in communications as communication using positive acknowledgment with retransmission.

Figure 113 represents an example of three simultaneously outstanding write request messages, with a single concatenated message that contains the responses for all the outstanding write requests. Concatenation is used in order to save messaging traffic.

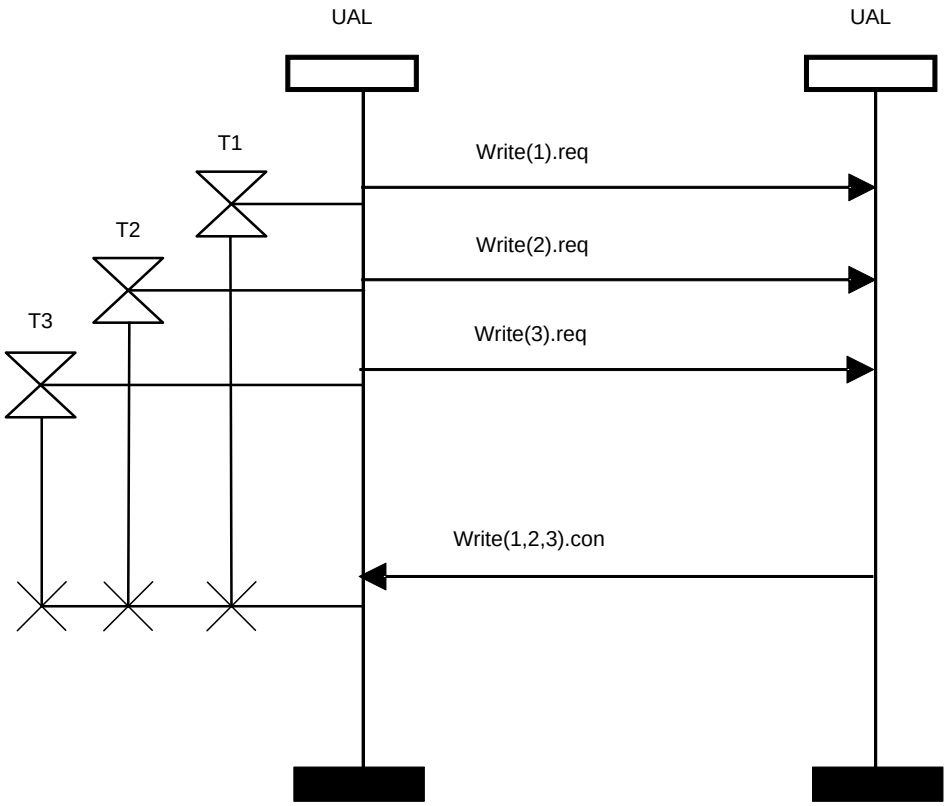


Figure 113 – A successful example of multiple outstanding requests, with response concatenation

12.12.4.2.2 Retries and timeout intervals

12.12.4.2.2.1 General

The method defined by IETF RFC 6298 shall be used for calculating an appropriate value for the retry timer-out interval (RTO). To compute the current RTO, a client shall maintain two state variables, $SRTT$ (smoothed round-trip time) and $RTTV$ (round-trip time variation), within the scope of a contract.

Until a round-trip time (RTT) measurement has been made for a segment sent between the client and server, the client should set RTO equal to 3 s.

When the first RTT measurement R is made, the client shall set:

$$SRTT = R$$

$$RTTV = R/2$$

$$RTO = SRTT + 4 \times RTTV$$

When a subsequent RTT is available, R is made. The client shall update the $RTTV$, $SRTT$, and RTO using the following calculations, where the recommended value for β is 0,25, and the recommended value for α is 0,125:

$$RTTV = (1 - \beta) \times RTTV + \beta \times |SRTT - R|$$

$$SRTT = (1 - \alpha) \times SRTT + \alpha \times R$$

$$RTO = SRTT + 4 \times RTTV$$

Whenever RTO is computed, the RTO shall be rounded based on the following rules:

If RTO < 1 s, set RTO to 1 s.

If RTO > 60 s, set RTO to 60 s.

Determination of timeout occurrence is a local matter. When a timeout has been determined to have occurred, exponential backoff shall be employed for consecutive timeouts by setting $RTO = RTO \times 2$ to send the retries. The maximum value of 60 s should be used to provide an upper bound to this doubling operation. Retries cease either when a response is received, or when the maximum retry limit is reached. The maximum retry permitted for a client request is indicated via an attribute of the UAP management object. The value selected for the maximum retry permitted is a local matter.

NOTE IETF RFC 6298 contains a recommendation regarding management of the TimeoutInterval timer.

12.12.4.2.2.2 Retries for unordered messages

Unordered messages are independent in that the order of responses may be received in a different order than the order in which the requests were sent. Accordingly, each request message times out and is retried independently.

Figure 114 is an example of how a timeout and retry of the second message in a sequence of three unordered messages, due to failure of the request to reach the server, are handled.

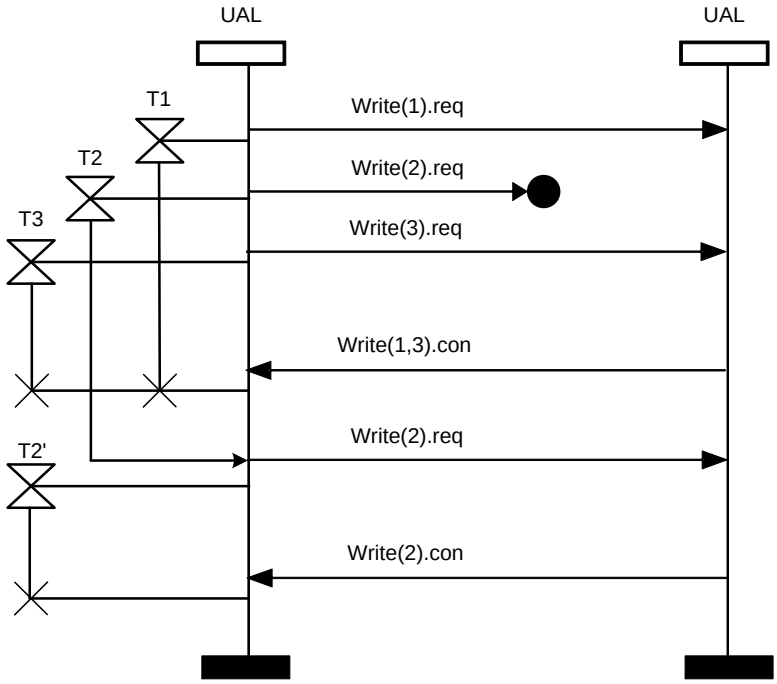


Figure 114 – An example of multiple outstanding unordered requests, with second write request initially unsuccessful

12.12.4.2.2.3 Retries for ordered messages

Ordered messages are order-dependent; that is, the order of responses may not be received in a different order than the order in which the requests were sent. Accordingly, if a later message receives a response before an earlier message, it indicates that the message for which no response was received shall be timed-out and retried.

Figure 115 is an example of how a timeout and retry of the second message in a sequence of three ordered messages, due to failure of the request to reach the server, are handled.

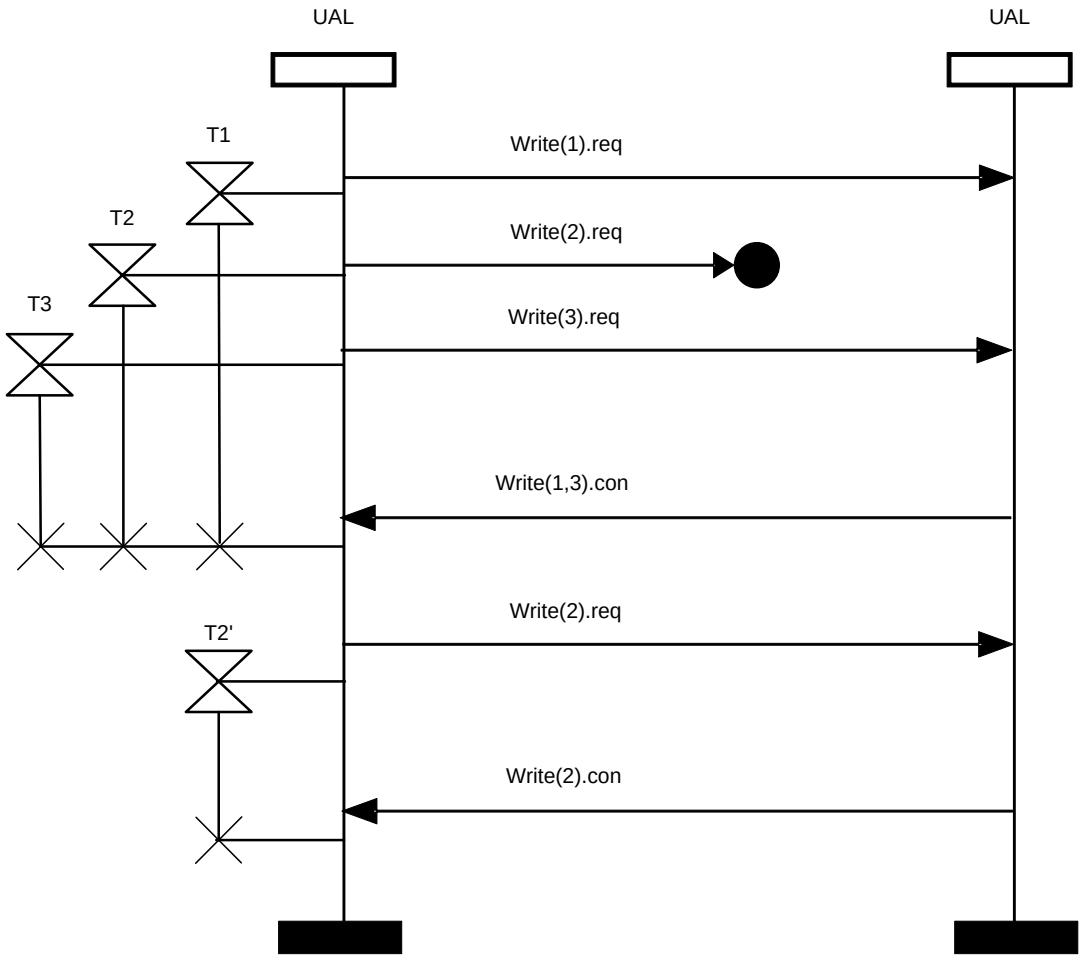


Figure 115 – An example of multiple outstanding ordered requests, with second write request initially unsuccessful

Ordered delivery only pertains to upload/download. The Max_Send_Window_Size for upload/download communication contracts shall be fixed at 1. As such, ordered message delivery is not supported by the lower layers of the protocol suite defined by this standard.

12.12.4.2.3 Flow control

Client/server communications are not application process rate controlled; rather, AL flow rate fairness is enforced by the application processes on a per-contract basis, in order to minimize the cost fairness of congestion on the network.

Max_Send_Window_Size (Table 26) for a contract is the maximum number of client requests that may be simultaneously awaiting a response within the scope of a contract. It is recommended (but not required) that clients use sequentially contiguous request identifiers. The value of Max_Send_Window_Size is established on a contract basis by the system manager. The OutstandingList represents the messages that have been sent, and that are currently awaiting a response.

The AvailableSendWindowSize represents the usable send window, that is, the set of client requests that may be sent, without violating the Max_Send_Window_Size, taking into consideration the number of messages contained in the OutstandingList. When the windows are empty, and CurrentSendWindowSize equals the Max_Send_Window_Size, the usable send window stretches from the last acknowledged client request for the next Max_Send_Window_Size number of requests, and represents the set of client requests that may be sent, without violating the Max_Send_Window_Size.

Applications shall initially set their value for their CurrentSendWindowSize to be one (1). RTT is then measured by the application for n complete transactions. If no timeout occurs during these transactions, and if the Max_Send_Window_Size has not been reached, the CurrentSendWindowSize shall be incremented by one (1). The value of CurrentSendWindowSize shall be equal the size of the CurrentSendWindowLimit +1.

NOTE This mechanism for increasing the window size is more conservative than that usually used to meet TCP congestion avoidance requirements.

As a response is received, the corresponding request moves from the sent window to the response completed window, and the size of AvailableSendWindowSize increases by one if the Max_Send_Window_Size has not been reached. If a timeout occurs awaiting a response, message loss or congestion is indicated. If this occurs, then:

- No additional message shall be placed into the OutstandingList until after the OutstandingList has first become empty.
- The CurrentSendWindowLimit shall be set to one (1).
- The messages that were in the OutstandingList at time of collapse shall be retried in order, according to the retry policies defined above. Retries use exponential backoff if the first retry does not succeed. Retries shall continue until either a response is received or the maximum number of retries has been met. When either of these conditions occurs, the message handling is considered complete, and the message shall be removed from the OutstandingList.

Client requests may continue again, building up the CurrentSendWindowLimit to the Max_Send_Window_Size value using the procedure described above.

EXAMPLE In an example of how the windows are used in a situation when there are no retries:

Let Max_Send_Window_Size be a constant, equal to the maximum number of simultaneously outstanding requests permitted by the contract. This limit is established by the system manager.

Let CurrentSendWindowSize be the variable that represents the number of simultaneously outstanding requests that exist for the contract at point in time t. CurrentSendWindowSize is a non-negative integer less than or equal to Max_Send_Window_Size.

Let UsedSendWindowSize be the variable that represents the number of simultaneously outstanding requests that are still awaiting responses.

AvailableSendWindowSize = CurrentSendWindowSize – UsedSendWindowSize.

Assume: Max_Send_Window_Size = 3

T1: Initialization: CurrentSendWindowSize = 1; UsedSendWindowSize = 0; AvailableSendWindowSize = 1;

T2: Message M_1 sent: CurrentSendWindowSize = 1; UsedSendWindowSize = 1; AvailableSendWindowSize = 0;

T3: Message M_1 response received: CurrentSendWindowSize = 1; UsedSendWindowSize = 0; AvailableSendWindowSize = 1;

T4: Message M_2 sent: CurrentSendWindowSize = 1; UsedSendWindowSize = 1; AvailableSendWindowSize = 0;

T5: Message M_3 response received: Current SendWindowSize = 2; UsedSendWindowSize = 0; AvailableSendWindowSize = 2.

The CurrentSendWindowSize has been incremented by one, since:

a) it has been at size 1, and 2 transactions have completed successfully,

b) CurrentSendWindowSize < Max_Send_Window_Size.

T6: Message M_4 sent: CurrentSendWindowSize = 2; UsedSendWindowSize = 1; AvailableSendWindowSize = 1;

T7: Message M_5 sent: CurrentSendWindowSize = 2; UsedSendWindowSize = 2; AvailableSendWindowSize = 0;

T8 Message M_4 and M_5 responses received : CurrentSendWindowSize = 2; UsedSendWindowSize = 0; AvailableSendWindowSize = 2;

T9: Message M_6 sent: CurrentSendWindowSize = 2; UsedSendWindowSize = 1; AvailableSendWindowSize = 1;

T10: Message M_7 sent: Current SendWindowSize = 2; UsedSendWindowSize = 2; AvailableSendWindowSize = 0;

T11: Message M_6 response received : Current Send Window Size = 3; UsedSendWindowSize = 1; AvailableSendWindowSize = 2.

The CurrentSendWindowSize has been incremented by one, since:

c) it has been at size 2, and 3 transactions have completed successfully,

d) CurrentSendWindowSize < Max_Send_Window_Size.

T12: Message M_8 sent: CurrentSend WindowSize = 3; UsedSendWindowSize = 2; Available SendWindowSize = 1;

T13: Message M_9 sent: CurrentSendWindowSize = 3; UsedSendWindowSize = 3; AvailableSendWindowSize = 0;

T14: Messages M_7 response received: CurrentSendWindowSize = 3; UsedSendWindowSize = 2; AvailableSendWindowSize = 1;

T15: Message M_{10} sent: CurrentSendWindowSize = 3; UsedSendWindowSize = 3; AvailableSendWindowSize = 0.

Figure 116 depicts a situation wherein the current send window has not yet built up to the maximum send window limit size. In this example, the maximum send window limit is three messages, one message in the outstanding list has been sent and is awaiting a response, and one message may be sent before the usable send window limit is reached.

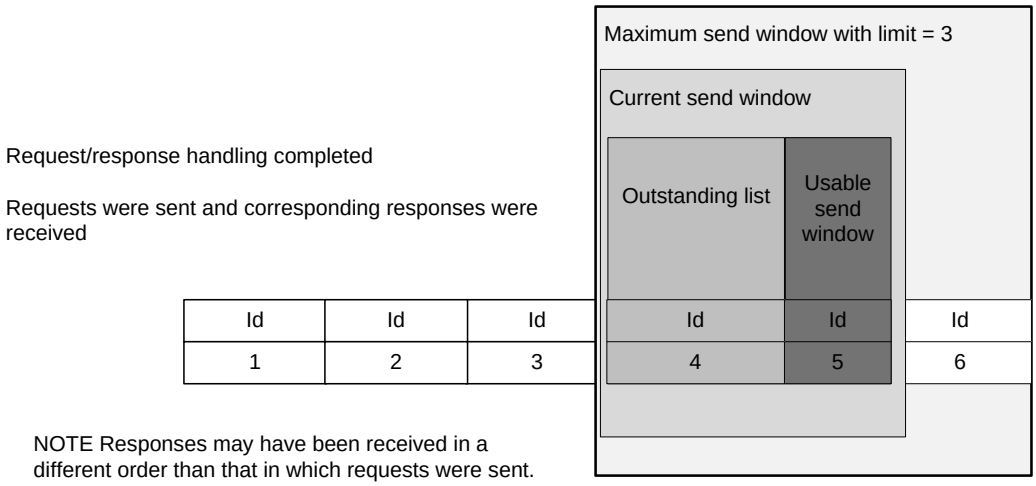


Figure 116 – Send window example 1, with current send window smaller than maximum send window

Figure 117 depicts a situation wherein the current send window has built up to the maximum send window limit size. In this example, the maximum send window limit is three messages, one message in the outstanding list has been sent and is awaiting a response, and two messages may be sent before the usable send window limit is reached.

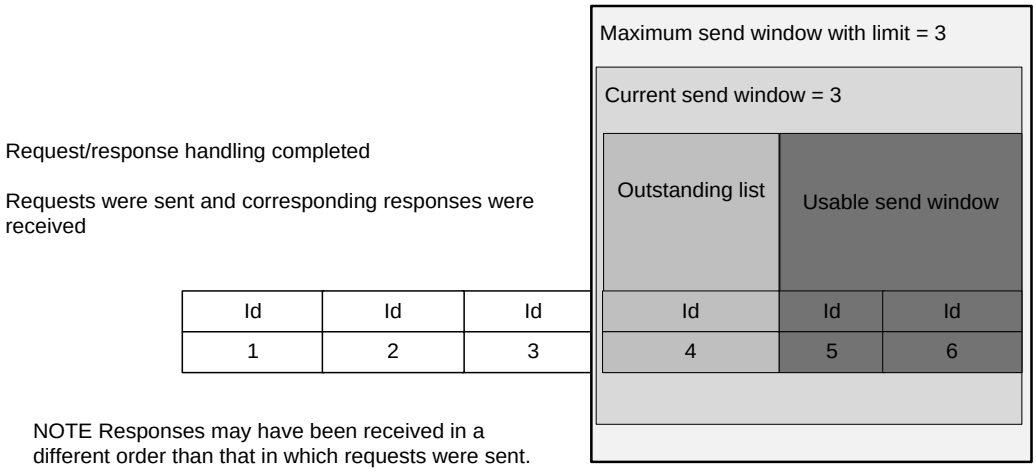


Figure 117 – Send window example 2, with current send window the same size as maximum send window, and non-zero usable send window width

Figure 118 depicts a situation wherein the current send window has built up to the maximum send window limit size. In this example, the maximum send window limit is three messages, and three messages have been sent and are awaiting responses.

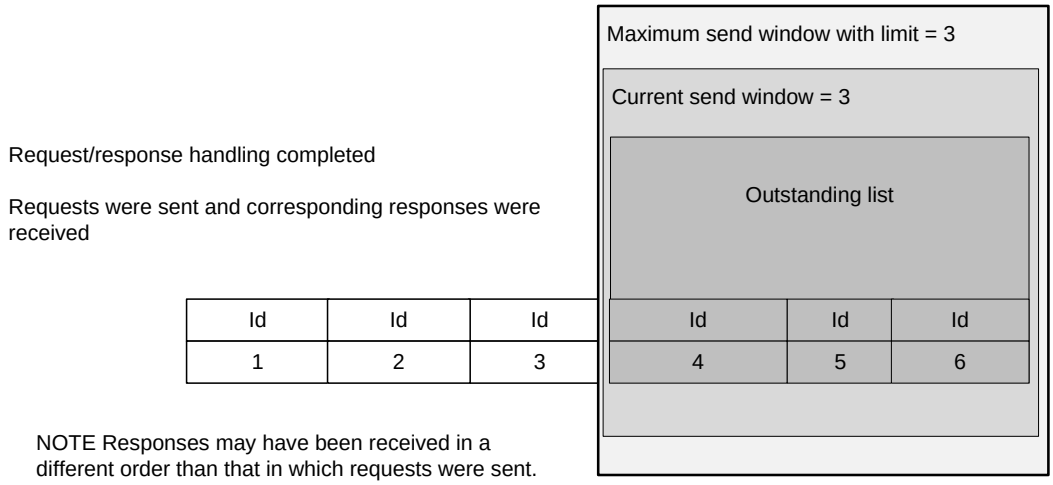


Figure 118 – Send window example 3, with current send window the same size as maximum send window, and usable send window width of zero

12.12.4.2.4 Probing for congestion

Some system configurations are more likely than others to incur message loss due to network congestion. In system configurations where congestion is more likely, an application may wish to regulate its AL service requests based on whether or not network congestion is present. To do this, an application may probe for congestion. To effect such a probe, the application may engage in a simple single message exchange.

NOTE 1 Probing is intended for diagnostic purposes only. A single message is used to ensure that the probes do not overload the network, and to ensure that the response to a probe is distinguishable.

The message request to use when probing shall be a non-concatenated read service. The read request and corresponding read response for the probe shall each fit within a single DL fragment. Any object attribute may be used as a probe; however, it is recommended (but not required) that the same object and attribute be used consistently for probing. For example, a standard attribute of the UAPMO may be used to probe UAPs, and a standard attribute of the DMAP DMO for probing the DMAP since those objects are required to be present in the corresponding applications.

If the probe timeout does not expire prior to reception of the response, then the application should assume that there is no congestion. However, if the response does not return before the retry timeout interval passes, this indicates a higher probability that network congestion is present. In this situation, the application process shall self-regulate its communication activities by setting its CurrentWindowSize to 1. See 12.12.4.2.3. If the application desires to send another congestion probe message, it may do so using exponential backoff as described in 12.12.4.2.2.1, but shall use a temporally-distinguishable request identifier for each message probe.

NOTE 2 Such distinction makes it possible for an application to compute RTT specific to congestion probing, and to make congestion decisions accordingly based on the congesting probing RTT data.

For example, a UAP in device X may issue a read service request for the required standard state attribute of the required UAPMO contained within the destination application in device Y. This read request may be treated as an application process-initiated congestion probe.

In the specific case of a download or upload, an application may probe for congestion. In these situations, congesting probing shall be performed as follows:

- To probe prior to commencing a download operation, the client probe shall be a read request to the UploadDownload attribute MaxDownloadSize.
- To probe for congestion during a download operation, the client probe shall be a read service request for the UploadDownload object's LastBlockDownloaded attribute.
- To probe prior to commencing an upload operation, the client probe shall be a read request to the UploadDownload attribute MaxUploadSize.
- To probe for congestion during an upload operation, the client probe shall be a read service request to the UploadDownload object's LastBlockUploaded attribute.

12.12.5 Communication service contract

A UAP makes a contact request to the local DMAP via an UAPME-n SAP in order to establish an agreement for a communication service needed by the UAP. If the need can be met, the DMAP provides a service contract identifier to the UAP that represents the agreement. The contract identifier is passed from the UAP to the ASL when it makes an ASL service request. This service contract ID is then used by the lower communication protocol suite to identify the layer-specific characteristics of the contract that have been established into the lower communication protocol suite layers by the local DMAP as part of establishing the service contract. The communication of information required from the UAP to the DMAP in order to acquire a service contract identifier is a device-internal matter, and hence not specified by this standard.

All communication contracts have a base set of information. Additional required information depends on the type of communication relationships desired. For example, a publish/subscribe relationship for periodic communication requires specification of the desired phase and period.

This standard does not specify how to determine the information needed by the UAP to specify the characteristics of a contract. For example, such information may be configured, such as the periodicity and phase to use in scheduled communication, or such information may be determined by the vendor of the device that contains the UAP.

Contract requests may be negotiated down by the system manager. UAP policies regarding the handling of negotiated down contracts, as well as policies regarding the handling of a declined contract, are outside the scope of this standard. See 6.3.11 for further information about the information that needs to be specified to request a contract.

The publishing period is represented by a signed 16-bit integer value. A positive value indicates a publication period as a multiple of 1 s (e.g., a value of 5 indicates a publishing period of 5 s; a value of 3 600 indicates a publishing period of 1 h). A negative value indicates publication on a fraction of 1 s (e.g., -4 indicates publish every ¼ s, -2 indicates publish every ½ s). A zero value indicates that no publishing should occur.

The periodicity selected should be based on the efficiency of the operation with this standard and the typical process practice.

DMAP knowledge of the destination TSAP port is not a requirement for creating a service contract, as contract establishment is concerned with resources for communication over the network conditions, whereas the destination port is used within the destination device, after the over-the-network communication has occurred.

NOTE Policies to retry establishment of a contract in the event of failure of contract establishment or revocation of a contract are behaviors of the device as a whole (as opposed to behaviors of a component within the device). Device-level behaviors are discussed in 6.3.11.2.4.2.

12.13 AL addressing

12.13.1 General

Certain information is required to address an object, an object's attribute, an element of an object's attribute (e.g., an element of a structure or an element of an array), or an object's method in native communications. Figure 119 represents the general addressing model for UAL process objects.

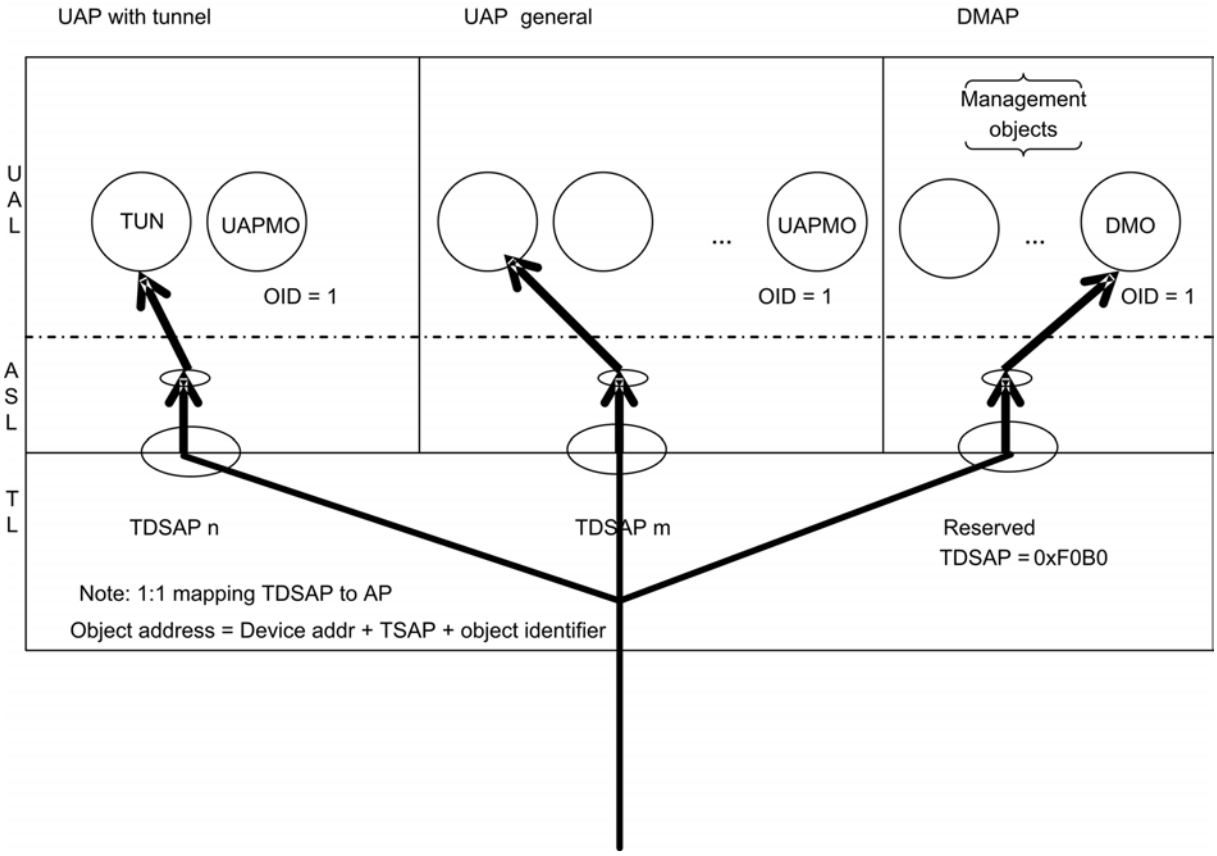


Figure 119 – General addressing model

12.13.2 Object addressing

An object is addressed in unicast communications by specifying:

- its containing device physical address;
- the TL TDSAP that is used to communicate with the unique UAL process that contains the object (that is, the TDSAP maps 1:1 to an application process);
- a T-port number that corresponds to a particular TDSAP; and
- the unique object identifier within the UAL process.

T-ports shall be assigned in consecutive order to TDSAPs, starting from the first available T-port. For example, TDSAP number 0 shall be correlated with the first T-port 0xF0B0. TDSAP number 1 shall be correlated with the second port, 0xF0B1, and so forth.

Particular TDSAPs, and their corresponding T-ports, are reserved by this standard so that they are well-known to all applications. Specifically, the DMAP in every device shall have the reserved transport port number 0xF0B0, which is associated with TDSAP number 0. The SMAP in a device shall have the reserved transport port number 0xF0B1, which is associated with TDSAP number 1. Devices that do not have an SMAP present shall not use T-port number 0xF0B1.

It is recommended that UAPs that are anticipated to have a large amount of messaging use the T-ports numbered 0xF0B2 through 0xF0BF, as they are represented in compressed form over the network, thus minimizing use of network resources, as well as RF congestion and device energy demand. See 11.4.4 for further details on TDSAPs and T-ports.

In order to minimize the encoding of application messages, it is recommended that object identifiers be allocated consecutively, starting at 1.

NOTE Object identifier 0 is reserved by this standard for the use of the application process management object contained within all application processes.

Multicast communication is not supported.

12.13.3 Object attribute addressing

An attribute of an object is addressed by specifying:

- the addressing of its containing object; and
- the unique attribute identifier within the object.

In order to minimize the encoding of application message attributes, it is recommended that attributes be allocated consecutively, starting at 1.

NOTE Attribute identifier 0 is reserved by this standard as a means of referring to an aggregate as a whole.

12.13.4 Object attribute addressing

12.13.4.1 General

Addressing for attributes is defined based on the type of attribute. This standard supports the following attribute types:

a) standard scalar types defined by this standard;

b) 1-origin singly-dimensioned homogeneous or heterogeneous arrays of elements of type a);

c) [1,1]-origin doubly-dimensioned arrays, where the first dimension indexes a homogeneous array of elements of type b).

Standard data structures defined by this standard are modeled and accessed as 1-origin singly-dimensioned heterogeneous arrays of elements. Thus access to the k^{th} member of a data structure, as enumerated in the declaration order of its member elements, is provided by accessing its k^{th} element as if the data structure were a 1-origin heterogeneous array.

NOTE In programmatic terms, this means that access to the k^{th} member of structure s, which programmatically might be referenced as s.memberName$_k$, is accessed as if it were $s[k]$, where k is the 1-origin ordinal index of the member in the containing declaration.

Elements of a doubly-dimensioned array that are themselves structures or arrays are accessible only by representing those individual elements as octet strings of uniform size.

12.13.4.2 Scalars

This standard supports access to attributes that are scalars of the following types:

- Boolean, mapped to Boolean8 or to Boolean1 when in a packed data structure;
- Integer, mapped to Integer8, Integer16, Integer32, Unsigned8, Unsigned16, Unsigned32, Unsigned64, Unsigned128, or to UnsignedN where $N < 16$ when in a packed data structure;
- Float, mapped to Float32 or Float64;
- VisibleString, mapped to VisibleStringN when N is fixed or determined by context;

- OctetString, mapped to OctetStringN when N is fixed or determined by context;
- BitString, mapped to BitStringN when N is fixed or determined by context;
- SymmetricKey, mapped to OctetString16 in this edition of this standard.

NOTE 1 Each BitString is represented as an integral number of octets or as an appropriate number of adjacent bits when in a packed data structure.

NOTE 2 See 12.22.3 on data types for the scalar types supported by this standard.

NOTE 3 OctetString and BitString provide a means for transparent conveyance of information that is unintelligible to the conveying protocol layer.

12.13.4.3 Structured protocol addresses treated as scalars

The following are also considered scalars when used in the data structures of this standard, even though their own defining standards specify a substructure for the item:

- IPv6Address, mapped to Unsigned128 to support simple numeric comparison;

NOTE 1 The substructure of this class of address is specified in IETF RFC 2460 and its related standards.

- EUI64Address, mapped to Unsigned64 to support simple numeric comparison;

NOTE 2 The substructure of this class of address is specified by the IEEE's Guidelines for 64-bit Global Identifier (EUI-64™).

- DL16Address, mapped to Unsigned16 to support simple numeric comparison.

In IEEE 802.15.4, the value 0xFFFF is the broadcast DL16Address, while any value in the range 0x0000..0x7FFD may be assigned to a DLE as a unicast DL16Address. However, this standard reserves the value 0 to indicate an unassigned DL16Address, so for this standard the range of unicast DL16Addresses is 0x0001..0x7FFF.

NOTE 3 IEEE 802.15.4 reserves the value 0xFFFE. IETF RFC4944 (6LoWPAN over IEEE 802.15.4) specifies that the range 0x80FF..0x9FFF is reserved for D-subnet-local multicast. Subclause 9.1.6.4 specifies that the range 0xA000..0x0AFFF is reserved by this standard for graph numbers used in source routes.

12.13.4.4 Singly-dimensioned arrays and standard data structures

This standard supports access to standard data structures and to arrays of either scalar elements or standard data structures.

Supported access to an array, or to a standard data structure not contained within an array, is as follows:

- A singly-dimensioned array or standard structure a may be accessed in its entirety by specifying access to member zero (e.g., an "index" value of 0, a[0]).
- A single member of a singly-dimensioned array or standard structure a may be accessed by identifying the 1-origin index of the desired member k , as specified in 12.13.4.1.
- A scalar member k of a standard structure member b of a standard structure a (e.g., $a.b.k$, where a and b are standard structures and k is a scalar supported by this standard).
- A singly-dimensioned array element b of a standard structure a may be accessed in its entirety by specifying access to member zero (e.g., $a.b$[0], where a is a standard structure and b is a singly-dimensioned array, and the array b is comprised either of scalars or standard structures as defined by this standard).
- A single element of a singly-dimensioned array b of standard structures that is a member of a standard structure a (e.g., $a.b[k]$, where a is a standard structure, b is a singly-dimensioned array comprised either of scalars or standard structures as defined by this standard, and k is the 1-origin index of the member of interest).

12.13.4.5 Singly-dimensioned arrays

This standard supports access to a singly-dimensioned array, whose individual members are either scalars or standard data structures as defined by this standard, as follows:

- A single element of a single dimension array, comprised of scalars (e.g., $a[k]$, where a specifies the array and k specifies the element in the array).
- A singly-dimensioned array of scalars or standard data structures may be accessed in its entirety (e.g., $a[0]$, where a specifies the array, and 0 specifies that access is to the entire array).
- An element of a singly-dimensioned array comprised of standard structures (e.g., $a[k][0]$, where a specifies the array, k specifies the array element that is the standard structure, and 0 specifies that access is to the entire member structure).
- A scalar member of a standard structure contained in a singly-dimensioned array (e.g., $a[k].j$, where a specifies the structure as defined by this standard, k specifies the array element that is the standard structure, and j specifies the member within that standard structure).
- A singly-dimensioned array contained as a member of a singly-dimensioned array (e.g., $a[k][0]$, where a is an array of standard structures, k specifies the element of the array, and 0 specifies that access is to the entire array).
- A member of a singly-dimensioned array of scalars or standard structures contained as a member of a singly-dimensioned array (e.g., $a[k][j]$, where a specifies the outer scope array, k specifies an element of that array that is itself an array, and j specifies the element of the inner scope array).

12.13.4.6 Doubly-dimensioned arrays

This standard supports access to a doubly-dimensioned array, consisting of a singly-dimensioned homogeneous array of singly-dimensioned homogeneous or heterogeneous arrays of scalars as defined by this standard, as follows:

a) a scalar element of a doubly-dimensioned array (e.g., $a[k][j]$);

b) a doubly-dimensioned array in its entirety (e.g., $a[0][0]$);

c) a row of a doubly-dimensioned array (e.g., $a[k][0]$); or

d) a column of a doubly-dimensioned array (e.g., $a[0][k]$).

NOTE Addressing form d) specifies a *slice* of the array, where the result is a singly-dimensioned array whose elements are the k^{th} member of each subarray. This slice mode of access enables selective access to any single member (element) of each component data structure in an array of identically-structured data structures.

12.13.5 Object method addressing

An object method is addressed by specifying

- the addressing of its containing object, and
- the object-unique index of the method identifier of the object.

12.14 Management objects

Standard management objects to manage the device as a whole are defined in this standard. These objects are defined in 6.2 and are accessed through a UAL-contained MP that may include, for example, a management object to support identification of the device, management objects for each layer of the communication protocol suite, and a management object to report alerts from the device.

NOTE Though each object tracks its own event and alarm conditions, the reporting of such conditions is specified by a single ARMO for the device as a whole. This object manages aspects including, but not limited to, the local alert reporting queue(s), the local timer(s) associated with retransmitting if an individual alert acknowledgment is not received, the local alert queue overflow handling, and requests for alarm recovery. See 6.2.7.2 for further details.

12.15 User objects

12.15.1 General

Standard UAP-containable objects are defined to enable interworkability across industries and segments. These objects may be industry-independent (that is, applicable across industries supported by this standard), or industry-dependent (that is, applicable to a particular industry supported by this standard, but not used across industries).

12.15.2 Industry-independent objects

12.15.2.1 General

The standard objects (UAPMO, ARO, UDO, Concentrator, Dispersion, Tunnel, and Interface) are applicable across industries supported by this standard.

12.15.2.2 UAP management object

12.15.2.2.1 General

There is exactly one addressable UAP management object (UAPMO) per UAP supported by the AL defined by this standard. The numeric object identifier of an object indicates a particular object instance. The numeric object identifier of the UAPMO in every UAP shall be fixed and shall have the value one (1). This object facilitates common management of application processes within a device. Attributes of this object are used to indicate such information as the version/revision of the application process and the logical status of the application process. For example, an attribute of the UAPMO indicates if the corresponding UAP is active or inactive.

NOTE 1 It is possible for a UAPMO to support management of a particular group (set) of objects within the UAP.

NOTE 2 Dynamic instantiation of UAPs is outside the scope of this standard.

12.15.2.2.2 Object attributes

A UAPMO has the attributes defined in Table 240.

Table 240 – UAP management object attributes *(1 of 2)*

Standard object type name: UAP management object (UAPMO)				
Standard object type identifier: 1				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
ObjectIdentifier	Object key identifier	Unique identifier for the object	Type: Unsigned16 Classification: Constant	N/A
UAP_ID	Object key identifier	Associated TLDE SAP	Type: Local matter (as defined by local TL) Classification: Constant	Local TDLE-SAP
UAP_TL_Port	Object key identifier	Associated T-port	Type: Unsigned16 Classification: Constant	NOTE 1 The specification of the UAP to its local TL is a local matter. NOTE 2 Transport defines the hexadecimal value set 0xF0B0+n (where n may range from 0..15) to specify the most compressed representation (into 4 bits) for communication.
Reserved for future use	0	—	—	—
VersionRevision	1	VersionRevision of the UAP	Type: VisibleString Max size: 64 octets Classification: Constant Accessibility: Read only	Human readable identification associated with the UAP management object NOTE The UAP vendor determines the content of this attribute.
State	2	Status of UAP	Type: Unsigned8 Classification: Static Accessibility: Read only Default value: 1 Named values: 0: inactive; 1: active; 2: failed	See Table 241
Command	3	Command to change the state of the UAP	Type: Unsigned8 Classification: Static Accessibility: Read/write Default value: 0 Named values: 0: none; 1: stop; 2: start; 3: soft reset; 4: hard reset	The value ‘none’ shall not be indicated in a write request. Soft reset shall preserve configuration/commissioning data. Hard reset returns application to factory default settings

Table 240 *(2 of 2)*

Standard object type name: UAP management object (UAPMO)				
Standard object type identifier: 1				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
MaxRetries	4	The maximum number of client request retries this application process will send in order to have a successful client/server communication	Type: Unsigned8 Classification: Static Accessibility: Read only Default value: 3 Valid range: 0..8 Classification: Static Accessibility: Read only	The number of retries sent for a particular message may vary by message based on application process determination of the importance of the message. For example, some messages may not be retried at all, and others may be retried the maximum number of times
Number of objects in the UAP including this UAPMO	8	Number of objects in the UAP including this UAPMO	Type: Unsigned8 Classification: Static Accessibility: Read only Default value: 1 Valid range: > 0	All UAPs are required to have a UAPMO, hence the default value is indicated as being 1 (one). The actual value of this attribute shall be the total number of objects contained in the UAP, including the UAPMO
Array of UAP contained objects	9	Identification of the objects and type contained in this UAP	Type: Array of ObjectIDandType Classification: Static Accessibility: Read only	See Table 271
Static_Revision_ Level	10	Revision level of the static data associated with all management objects	Type: Unsigned16 Classification: Static Accessibility: Read only Default value: 0	Revision level is incremented each time a static attribute value of any object contained in this UAP is changed
Reserved for future use by this standard	5..7 11..63	—	—	*N* octets of presently undefined content

12.15.2.2.3 State table for UAP management object

Table 241 describes the state table for the UAP management object.

Table 241 – State table for UAP management object

Transition	Current state	Event(s)	Action(s)	Next state
T1	Inactive	Write(Command, Start)	Write.rsp(success)	Active
T2	Active	Write(Command,Stop)	Write.rsp(success)	Inactive
T3	Inactive	Write(Command,Stop)	Write.rsp(success)	Inactive
		Write(any Reset command)	Write.rsp(operationAccepted)	
T4	Active	Write(Command,Start)	Write.rsp(success)	Active
		Write(any other command)	Write.rsp(objectStateConflict)	
T5	Inactive	Write(Command,Start)	Write.rsp(failed) Note: Fails to start	Failed
T6	Active	Application problem	N/A	Failed
T7	Failed	Write(any Reset command)	Write.rsp(operationAccepted)	Inactive
T8	Failed	Write(any command other than Reset)	Write.rsp(objectStateConflict)	Failed

Figure 120 shows the UAP management object state diagram.

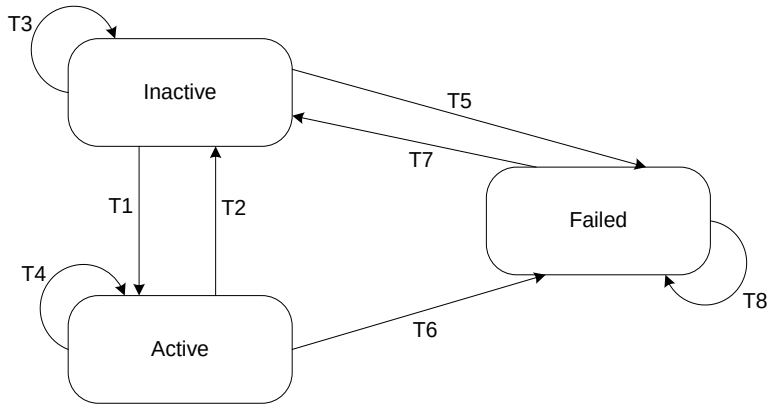


Figure 120 – UAP management object state diagram

12.15.2.2.4 Standard object methods

A UAP management object has methods as defined in Table 242.

Table 242 – UAP management object methods

Standard object type name: UAP management object		
Standard object type identifier: 1		
Method name	**Method ID**	**Method description**
Null	0	Reserved by standard for future use
Reserved for future use by this standard	0..127	These method identifiers are reserved for future use by this standard
Implementation-specific use	128..255	These method identifiers are available for implementation-specific use

12.15.2.3 Alert-receiving object

12.15.2.3.1 General

There may be up to four alert-receiving objects in a device, one per alert reporting category. These alert-receiving objects may receive more than one category of alert report. Categories of alert reports received by alert objects shall be unique; that is, if one alert-receiving object is receiving alerts of category X from the ASL, no other alert objects in the device may also receive alerts of category X from the ASL. These alert-receiving objects may be contained in the same or different processes (e.g., an alert-receiving object for security alerts may be contained in one application process, while another for process alerts may be contained in another application process).

NOTE Further separation of alerts, or consolidation and re-reporting of alerts, if necessary, is an application process local matter, outside the scope of the AL specification.

12.15.2.3.2 Object attributes

An alert-receiving object may receive alerts from one or more alert-reporting sources. The object has the attributes defined in Table 243.

Table 243 – Alert-receiving object attributes

Standard object type name: Alert-receiving object				
Standard object type identifier: 2				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
ObjectIdentifier	Object key identifier	Unique identifier for the object	Type: Unsigned16	N/A
			Classification: Constant	
			Valid range: > 0	
Reserved for future use	0	—	—	—
Categories	1	BitString of alert categories indicating which object instance supports receiving	Type: BitString	N/A
			Classification: Static	
			Accessibility: Read only	
			Default value: 0	
			Named indices: 0: device alerts; 1: communication alerts; 2: security alerts; 3: process alerts; 4..7: reserved for future use by this standard	
Errors	2	Count of reports received for a category that the receiving object indicated was not supported	Type: Unsigned16	Wraps to 0 when maximum value is reached
			Classification: Dynamic	
			Accessibility: Read only	
			Default value: 0	
Reserved for future use by this standard	3..63	—	—	N octets of presently undefined content

12.15.2.3.3 State table for AlertReport handling

Table 244 indicates the states for handling reception of an AlertReport.

Table 244 – State table for handling an AlertReport reception

Transition	Current State	Event(s)	Action(s)	Next State
T1	Ready	AlertReport.ind received	Note processing alert report from device X	Handle individual alert in report
T2	Handle individual alert in report	Check category	Valid category: Acknowledge alert report, and process it	Ready
			Invalid category: Increment the alert-receiving object instances value of its Errors attribute	

Figure 121 shows the state diagram for alert reception.

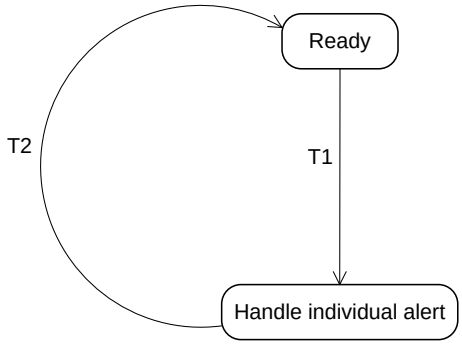


Figure 121 – Alert report reception state diagram

Figure 122 shows one example of alert reporting from multiple devices sources to multiple alert-receiving objects contained in a single UAP of a single sink device.

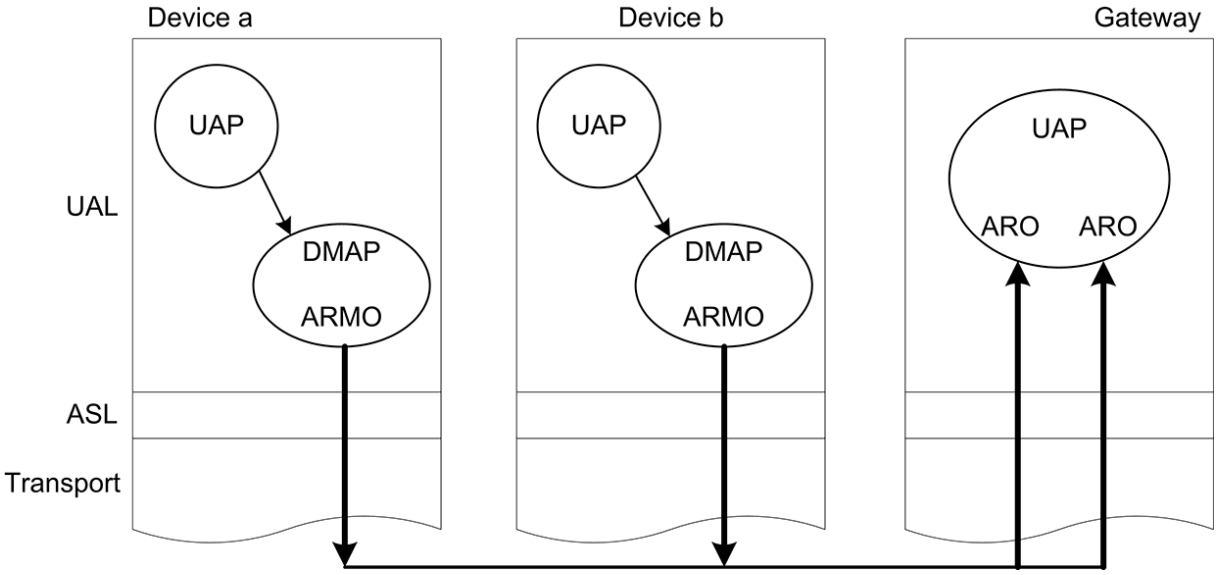


Figure 122 – Alert-reporting example

12.15.2.3.4 Standard object methods

An AlertReceiving object has the methods defined in Table 245.

Table 245 – AlertReceiving object methods

Standard object type name: AlertReceiving object		
Standard object type identifier: 2		
Method name	**Method ID**	**Method description**
Null	0	Reserved by standard for future use
Reserved for future use by this standard	0..127	These method identifiers are reserved for future use by this standard
Implementation-specific use	128..255	These method identifiers are available for implementation-specific use

12.15.2.4 UploadDownload object

12.15.2.4.1 General

An UploadDownload object is used for either uploading or downloading information to a device. The UploadDownload object may be used to support operations such as downloading a new version of operating firmware or downloading new UAP-contained code or UAP-required bulk data. The UploadDownload object maintains revision control information to indicate what was downloaded or what is available for upload (or both).

An UploadDownload object is likely to support upload or download for a single semantic set of information. An UploadDownload object shall support only one upload or download operation at a time.

A process may have zero or more UploadDownload object instances. Multiple UploadDownload object instances are required if more than one semantic set of information is needed to upload or download its required content.

NOTE 1 The local effect of an application process upload or download (e.g., the creation of new network-visible objects as a result of a download) is a local matter, outside the scope of this standard.

NOTE 2 UploadDownload objects are usable to support upload operations such as the upload of statistical or historical information from the device for analysis. An UploadDownload object is usable to update software/firmware in the target device.

NOTE 3 Support of multicast download is a subject of future standardization. To support multicast loads to a specific set of devices, a configuration tool is currently envisioned to be used to configure the multicast address/device/object relationships for the objects in the multicast set.

12.15.2.4.2 Object attributes

An UploadDownload object has the attributes defined in Table 246. Attributes are included in this object type in order to provide application-level communication timing guidance to the client that is communicating with the UploadDownload object.

NOTE Further guidance to the client, such as regarding tuning of communication timing (for example, related to network communication delays due to the topology of the messaging graph traversed, potential queuing delays, etc.), usable to tune client application behavior, is transparent to an application process, and hence the application itself is unable to provide complete guidance.

Table 246 – UploadDownload object attributes *(1 of 4)*

Standard object type name: UploadDownload object				
Standard object type identifier: 3				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
ObjectIdentifier	Object key identifier	Unique identifier for the object	Type: Unsigned16 Classification: Constant Valid range: > 0	N/A
Reserved for future use	0	—	—	—
OperationsSupported	1	Indicates if this object supports uploads, downloads, or both	Type: Unsigned8 Classification: Constant Accessibility: Read only Named values: 0: defined size unicast upload only; 1: defined size unicast download only; 2: defined size unicast upload and unicast download; 3..15: reserved for future use by this standard	N/A
Description	2	Human readable identification of associated content	Type: VisibleString SIZE (0..64) Classification: Static Accessibility: Read only	
State	3	State of the UploadDownload Object instance	Type: Unsigned8 Classification: Dynamic Accessibility: Read only Default value: 0 Named values: 0: idle; 1: downloading; 2: uploading; 3: applying; 4: DLComplete; 5: ULComplete; 6: DLError; 7: ULError	See state table below
Command	4	Action command to this object	Type: Unsigned8 Classification: Non-cacheable Accessibility: Read/write Named values: 0: reset; 1: apply (used for Download only); 2..15: reserved for future use by this standard	See Table 254

Table 246 *(2 of 4)*

Standard object type name: UploadDownload object				
Standard object type identifier: 3				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
MaxBlockSize	5	Maximum size of a block which can be accepted for a download, or provided for an upload	Type: Unsigned16	Unit: octets The value shall not exceed the maximum amount of data that can be conveyed in a single APDU per the communication contract. Additionally, space in the APDU shall be left for service related encoding. Block sizes conveyed may be smaller than this value, but shall not be larger
			Classification: Static	
			Accessibility: Read only	
			Default value: 1 to (MaxNPDUsize – Max TL header size – max(sizeof (additional coding of AL UploadData service request), additional coding of sizeof(AL DownloadData service response))	
			Valid range: 0..maximum size for data in an APDU	
MaxDownloadSize	6	Maximum size available for download as a whole	Type: Unsigned32	Unit: octets
			Classification: Static	
			Accessibility: Read only	
			Default value: 0	
MaxUploadSize	7	Size available for Upload	Type: Unsigned32	Unit: octets
			Classification: Static	
			Accessibility: Read only	
			Default value: 0	
DownloadPrepTime	8	Time required, in seconds, to prepare for a download	Type: Unsigned16	Time required between sending the StartDownload response till the object can handle a DownloadData
			Classification: Static	
			Accessibility: Read only	
			Default value: 0	
DownloadActivationTime	9	Time in seconds for the object to apply newly downloaded content	Type: Unsigned16	N/A
			Classification: Static	
			Accessibility: Read only	
			Default value: 0	
UploadPrepTime	10	Time required, in seconds, to prepare for an upload	Type: Unsigned16	Time from sending the StartUpload response till the object can accept an UploadData
			Classification: Static	
			Accessibility: Read only	
			Default value: 0	

Table 246 *(3 of 4)*

Standard object type name: UploadDownload object				
Standard object type identifier: 3				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
UploadProcessingTime	11	Typical time in seconds for this application object to process a request to upload a block	Type: Unsigned16 Classification: Static Accessibility: Read only Default value: 0	This information is intended to allow a client of an Upload operation to tune its upload related messaging to correspond to the operation of the particular UploadDownload object instance. For example, a client may use this time to help determine its timeout/retry policy, or to determine when to invoke a method on the object instance
DownloadProcessingTime	12	Typical time in seconds for this application object to process a downloaded block	Type: Unsigned16 Classification: Static Accessibility: Read only Default value: 0	This information may be used by a client of a Download operation to tune its download related messaging to correspond to the operation of the particular UploadDownload object instance. For example, a client may use this time to help determine its timeout/retry policy, or to determine when to invoke a method on the object instance
CutoverTime	13	Time (in seconds) specified to apply the download content	Type: TAINetworkTime Classification: Static Accessibility: Read Write Initial default value : 0	Downloaded content will be applied at the time specified by this attribute
LastBlockDownloaded	14	Number of last block successfully downloaded	Type: Unsigned16 Classification: Static Accessibility: Read only Default value : 0	Updated when an execute response to a DownloadData method is returned. Block number counting shall start at 1 (one). See 12.15.2.4.5.3
LastBlockUploaded	15	Number of last block successfully uploaded	Type: Unsigned16 Classification: Static Accessibility: Read only Default value : 0	Updated when an execute response to an UploadData method is returned. Block number counting shall start at 1 (one). See 12.15.2.4.5.3

Table 246 *(4 of 4)*

Standard object type name: UploadDownload object				
Standard object type identifier: 3				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
ErrorCode	16	Upload or Download error	Type: Unsigned8 Classification: Static Accessibility: Read only Default value : 0 Named values: 0: noError; 1: timeout; 2: clientAbort; 18: InconsistentContent; 27: InsufficientDevice Resources; 3..17, 19..26, 28..63: reserved for future use by this standard; 64..255: manufacturerSpecific	Updated when there is an error in uploading or downloading to this object. The error is cleared when the object transitions out of the error state to the idle state. Use InconsistentContent to indicate that the device did not cutover as scheduled due to problem with download payload. Use InsufficientDevice Resources to indicate that the download could not be completed due to lack of memory or other resources
Reserved for future use by this standard	16..63	—	—	—
This standard does not prescribe product lifecycle management or versioning policies. Description may be used to indicate interchangeability of versions, or to identify features/fixes/software builds. The maximum value of any additional coding for the application coding for this standard is 9 octets. Implementers may wish to consult IETF RFC 2348 regarding recommendations for a maximum size of PDUs.				

12.15.2.4.3 Standard object methods

Initiation of an upload or download bulk data transfer requires first reaching an agreement between the corresponding application objects to participate in the data transfer.

Any additional coordination required to ensure the readiness of the responding device to accept an upload or download request is the responsibility of the UAL process that will be starting a bulk transfer operation.

Client/server messaging to access coordination information from an attribute or set of attributes of the UploadDownload object may be used to support this coordination activity. Specifically, a read request may be used in advance of starting a bulk transfer in order for a client to collect bulk transfer communication-related information specific to an UploadDownload object instance. Once agreement is reached, the client application controls when the data is provided (for a download) to the UploadDownload object, or requested (for an upload) from the UploadDownload object. When transfer is complete, the client indicates the transfer has ended. Additionally, the client may close the transfer if it determines that the entire data transfer should not be completed.

The serving application may request the data transfer be aborted if it determines that the data transfer cannot or should not be completed.

A serving application making a decision to abort based on lack of communication from the client should at least allow for the default standard retries and retry timing policy for the client/server communication policy in order to establish an appropriate timeout.

As with other application communications, it is required that transmission bandwidth be allocated by a communication service contract in order to support a bulk data transfer. Bandwidth for bulk data transfer is not considered dedicated bandwidth as used for periodic messaging, but rather is considered shared bandwidth as used for aperiodic messaging. Use of shared bandwidth among all users of shared bandwidth by a device is dependent on a combination of overall contract priority and message priority. Contract priority is defined by the system manager. Message priority is defined by the application process.

NOTE 1 Any required coordination or sequencing of multiple images to different UploadDownload objects is the responsibility of the host application process. Different uploadable or downloadable images necessitate separate UploadDownload object instances.

NOTE 2 The semantics and syntax of the content and use of uploaded or downloaded information are outside the scope of this standard. The resulting activity in the application process of the device providing upload data or accepting download data, other than updating the UploadDownload object itself, is a local matter, and hence is outside the scope of this standard.

NOTE 3 A proxy application within the device is one way for a single device to process the download multiple times.

Upload from or download to a single device uses a unicast protocol; that is, the upload or download content is sent from/to a single UploadDownload object within a single device.

NOTE 4 File content and/or naming conventions, if applicable to an upload or download are outside the scope of this standard.

An UploadDownload object has the methods defined in Table 247.

Table 247 – UploadDownload object methods

Standard object type name: UploadDownload object		
Standard object type identifier: 3		
Method name	**Method ID**	**Method description**
Null	0	Reserved by standard for future use
StartDownload	1	This method is used by a client to reach an agreement with an UploadDownload object to participate in a download for which the client will be providing the data, one block at a time
DownloadData	2	This method is used by a client to provide data to an UploadDownload object for an agreed download operation
EndDownload	3	This method is used by a client to terminate a download operation that either has completed successfully, or which the client wishes to abort
StartUpload	4	This method is used by a client to reach an agreement with an UploadDownload object to participate in an upload for which the client will be requesting the data, one block at a time
UploadData	5	This method is used by a client to request data from an UploadDownload object for an agreed upload operation
EndUpload	6	This method is used by a client to terminate an upload operation that either has completed successfully, or that the client wishes to abort
Reserved for future use by this standard	7..127	These method identifiers are reserved for future use by this standard
Implementation-specific use	128..255	These method identifiers are available for implementation-specific use
The approach used for upload and download has roots in the experiences of multiple accepted standards, including but not necessarily limited to Fieldbus Foundation, Device Net International, IETF RFC 1350 and IETF RFC 2347. Attributes of the UploadDownload object provide application-level information to assist in timeout interval determination by a client, hence IETF RFC 2349 is not followed. Acknowledgment retry as proposed in IETF RFC 2347 is not adopted for the following reasons: • The use cases driving this standard, and the agreed set of technical requirements this standard was to meet, have the vast bulk of communication being publish/subscribe, with very limited use of upload/download. • The upload/download operations that have been defined are not time-critical. • The client application receives feedback from the server if the server is getting duplicates, and can elect to terminate the operation. • The server application is aware of when it is sending error messages back to the client, and is able to elect to abort the operation.		

12.15.2.4.4 StartDownload method

12.15.2.4.4.1 General

Table 248 describes the StartDownload method of the UploadDownload object.

Table 248 – UploadDownload object StartDownload method

<table>
<tr><th colspan="5">Standard object type name: UploadDownload object</th></tr>
<tr><th colspan="5">Standard object type identifier: 3</th></tr>
<tr><th>Method name</th><th>Method ID</th><th colspan="3">Method description</th></tr>
<tr><td rowspan="11">StartDownload</td><td>1</td><td colspan="3">A client uses the StartDownload method to indicate to an UploadDownload object instance that it desires to download the object</td></tr>
<tr><td colspan="4">Input arguments</td></tr>
<tr><td>Argument number</td><td>Argument name</td><td>Argument type</td><td>Argument description</td></tr>
<tr><td>1</td><td>BlockSize</td><td>Unsigned16</td><td>The size of a block of data in octets that will be downloaded</td></tr>
<tr><td>2</td><td>DownloadSize</td><td>Unsigned32</td><td>The total size of data to be downloaded in octets</td></tr>
<tr><td>3</td><td>DownloadMode</td><td>Unsigned8</td><td>The desired mode of operation</td></tr>
<tr><td colspan="4">Output arguments</td></tr>
<tr><td>Argument number</td><td>Argument name</td><td>Argument type</td><td>Argument description</td></tr>
<tr><td colspan="4">None</td></tr>
</table>

12.15.2.4.4.2 Method description

A client uses the StartDownload method to indicate to an UploadDownload object instance that it desires to download the object, and to specify the parameters of the download in the input argument list. The UploadDownload object may accept or reject the download, indicating one or the other outcome via the output argument list.

If an UploadDownload object accepts to participate in a download operation, it shall not accept another download operation, or an upload operation until the download operation in process has been terminated or the object has been reset.

12.15.2.4.4.3 Input arguments

Input arguments include:

- BlockSize, which indicates the size of a block of data in octets. All blocks shall have the same size, except that the last block of a download may contain a smaller positive number of octets.
- DownloadSize, which represents the total size of data to be downloaded, in octets.
- DownloadMode, which indicates the operational mode desired. The valid value for this argument indicates unicast and is represented by a value of zero.

12.15.2.4.4.4 Output arguments

There are no output arguments for this method.

12.15.2.4.4.5 Response codes

The following feedback codes are valid for this method:

- operationAccepted;
- invalidBlockSize;
- invalidDownloadSize;
- unexpectedMethodSequence;

- insufficientDeviceResources;
- deviceHardwareCondition; and
- those that are vendor-defined.

12.15.2.4.5 DownloadData method

12.15.2.4.5.1 General

Table 249 describes the DownloadData method of the UploadDownload object.

Table 249 – UploadDownload object DownloadData method

Standard object type name: UploadDownload object				
Standard object type identifier: 3				
Method name	**Method ID**	**Method description**		
DownloadData	2	A client uses the DownloadData method to provide data to an UploadDownload object that has agreed to be downloaded		
	Input arguments			
	Argument number	**Argument name**	**Argument type**	**Argument description**
	1	BlockNumber	Unsigned16	BlockNumber being downloaded
	2	Data	OctetString	The data for the block being downloaded. The maximum size of this string may vary, such as it may differ for different destination UploadDownload objects
	Output arguments			
	Argument number	**Argument name**	**Argument type**	**Argument description**
	1	CurrentBlockNumber	Unsigned16	This argument is present if the serviceFeedbackCode indicates either blockout of sequence or duplicate

12.15.2.4.5.2 Method description

StartDownload is used first to have the UploadDownload object agree to the download.

Data is sent one block at a time, sequentially from the lowest numbered block to the highest numbered block using the DownloadData method. Only one DownloadData method invocation may be outstanding at a time; for example, if DownloadData for block n has been invoked, DownloadData for block n + 1 shall not be invoked until a successful response containing the output arguments for the download of block n has been received by the client.

The UploadDownload object may indicate that it needs to abort via output argument MethodStatus.

If a client of an upload or download operation is issuing multiple data transfer method invocations for the same block, it may be due to either a network-related problem (e.g., the request is not reaching the server) or a problem at the server device. In this situation, the client may employ the appropriate operation end method (EndDownload or EndUpload) to terminate the operation.

If a client of an upload or download receives multiple dataSequenceError responses, it may be due either to network-related problems (for example, loss of a method invocation response), or problems at the server device. In this situation, the client may employ the appropriate operation end method (EndDownload or EndUpload) to terminate the operation.

Correspondingly, if an UploadDownload object has sent multiple dataSequenceError responses, it may infer that there are either network-related problems or problems at the client device and may elect to abort the operation. If an UploadDownload object indicates operation abort, and this abort is lost over the network, the response sent to a subsequent data method (DownloadData or UploadData) or end method (EndDownload or EndUpload) indicates that the object is no longer participating in an upload or download operation with this client by sending a response indicating unexpectedMethodSequence.

12.15.2.4.5.3 Input arguments

Input arguments include:

- BlockNumber, which is the number of the block for which data is provided, where the count of block numbers starts at 1 (one); and
- Data, which represents the data for the block being downloaded.

12.15.2.4.5.4 Output arguments

This current BlockNumber argument is present if the serviceFeedbackCode indicates either blockout of sequence or duplicate. The argument indicates the last BlockNumber received. The intent is to permit the client to resolve an out-of-sequence or duplicate block reception error without aborting the download operation.

12.15.2.4.5.5 Response codes

The following feedback codes are valid for this method:

- success;
- invalidBlockNumber;
- blockDataError (e.g., wrong block size; content problem);
- unexpectedMethodSequence;
- insufficientDeviceResources;
- deviceHardwareCondition;
- operationAborted;
- dataSequenceError (e.g., duplicate);
- timingViolation; and
- those that are vendor-defined.

12.15.2.4.6 EndDownload method

12.15.2.4.6.1 General

Table 250 describes the EndDownload method of the UploadDownload object.

Table 250 – UploadDownload object EndDownload method

<table>
<tr><th colspan="5">Standard object type name: UploadDownload object</th></tr>
<tr><th colspan="5">Standard object type identifier: 3</th></tr>
<tr><th>Method name</th><th>Method ID (non-negative)</th><th colspan="3">Method description</th></tr>
<tr><td rowspan="8">EndDownload</td><td>3</td><td colspan="3">A client uses the EndDownload method to indicate that the download is terminating</td></tr>
<tr><td colspan="4">Input arguments</td></tr>
<tr><td>Argument number</td><td>Argument name</td><td>Argument type</td><td>Argument description</td></tr>
<tr><td>1</td><td>Rationale</td><td>Unsigned8</td><td>This argument indicates the client’s reason for terminating the download operation</td></tr>
<tr><td colspan="4">Output arguments</td></tr>
<tr><td>Argument number</td><td>Argument name</td><td>Argument type</td><td>Argument description</td></tr>
<tr><td colspan="4">None</td></tr>
</table>

12.15.2.4.6.2 Method description

A client uses the EndDownload method to indicate that the download operation is terminating. Termination may occur, for example, if the download has completed, or if the client has elected to terminate the download operation.

EndDownload may be sent from a client that is presently engaged in a download operation, as agreed by the StartDownload method.

12.15.2.4.6.3 Input arguments

The Rationale argument indicates the client’s reason for terminating the download operation. The value used shall be from the following set:

- 0: download completed successfully; or
- 1: client abort.

12.15.2.4.6.4 Output arguments

There are no output arguments for this method.

12.15.2.4.6.5 Response codes

The following feedback codes are valid for this method:

- success;
- operationIncomplete;
- unexpectedMethodSequence;
- timingViolation; and
- those that are vendor-defined.

12.15.2.4.7 StartUpload method

12.15.2.4.7.1 General

Table 251 describes the StartUpload method of the UploadDownload object.

Table 251 – UploadDownload object StartUpload method

Standard object type name: UploadDownload object				
Standard object type identifier: 3				
Method name	**Method ID**	**Method description**		
StartUpload	4	A client uses the StartUpload method to indicate to an UploadDownload object instance that it desires to upload data from the object.		
	Input arguments			
	Argument number	**Argument name**	**Argument type**	**Argument description**
	1	DownloadMode	Unsigned8	The desired mode of operation
	Output arguments			
	Argument number	**Argument name**	**Argument type**	**Argument description**
	1	BlockSize	Unsigned16	The size of a block of data in octets
	2	UploadSize	Unsigned32	The total size of the data to be uploaded in octets

12.15.2.4.7.2 Method description

A client uses the StartUpload method to indicate to an UploadDownload object instance that it desires to upload data from the object. The UploadDownload object may accept or reject the upload, indicating the outcome via the output argument list.

If an UploadDownload object accepts to participate in an upload operation, it shall not accept another upload operation or a download operation until the upload operation in process has been terminated or the object has been reset.

12.15.2.4.7.3 Input arguments

Input arguments include:

- DownloadMode, which specifies the desired mode of operation. The valid value for this argument indicates unicast and is represented by a value of zero.

12.15.2.4.7.4 Output arguments

Output arguments include:

- BlockSize, which is the size of a block of data in octets. All blocks shall have the same size, except that the last block of an upload may contain a smaller positive number of octets.
- UploadSize, which indicates the size of the data to be uploaded, in octets.

12.15.2.4.7.5 Response codes

The following feedback codes are valid for this method:

- success;
- unexpectedMethodSequence;
- insufficientDeviceResources;
- deviceHardwareCondition;
- those that are vendor-defined.

12.15.2.4.8 UploadData method

12.15.2.4.8.1 General

Table 252 describes the UploadData method of the UploadDownload object.

Table 252 – UploadDownload object UploadData method

<table>
<tr><th colspan="5">Standard object type name: UploadDownload object</th></tr>
<tr><th colspan="5">Standard object type identifier: 3</th></tr>
<tr><th>Method name</th><th>Method ID</th><th colspan="3">Method description</th></tr>
<tr><td rowspan="8">UploadData</td><td>5</td><td colspan="3">A client uses the UploadData method to acquire data from an UploadDownload object that has agreed to be uploaded</td></tr>
<tr><td colspan="4">Input arguments</td></tr>
<tr><td>Argument number</td><td>Argument name</td><td>Argument type</td><td>Argument description</td></tr>
<tr><td>1</td><td>BlockNumber</td><td>Unsigned16</td><td>The number of the block for which data is requested</td></tr>
<tr><td colspan="4">Output arguments</td></tr>
<tr><td>Argument number</td><td>Argument name</td><td>Argument type</td><td>Argument description</td></tr>
<tr><td>1</td><td>Data</td><td>OctetString</td><td>This argument contains the data for the requested block. This argument is present if and only if the serviceFeedbackCode indicates success.
The maximum size of this may vary by UploadDownload object instance being uploaded</td></tr>
</table>

12.15.2.4.8.2 Method description

A client uses the UploadData method to acquire data from an UploadDownload object which has agreed to be uploaded.

The StartUpload is used first to have the UploadDownload object agree to the upload. Data is requested one block at a time, sequentially from the lowest numbered block to the highest numbered block. Only one UploadData method invocation may be outstanding at a time. For example, if UploadData for block n has been invoked, UploadData for block n + 1 shall not be invoked until the corresponding successful response containing the output arguments has been received by the client.

The UploadDownload object may indicate that it needs to abort via an output argument.

12.15.2.4.8.3 Input arguments

The BlockNumber argument specifies the number of the block for which data is requested. Block number counting shall start at 1 (one).

12.15.2.4.8.4 Output arguments

The Data argument contains the data for the requested block. This argument is present if and only if the serviceFeedbackCode indicates success.

12.15.2.4.8.5 Service feedback codes

The following feedback codes are valid for this method:

- success;
- unexpectedMethodSequence;

- insufficientDeviceResources;
- deviceHardwareCondition;
- operationAborted;
- dataSequenceError (e.g., duplicate, invalid block number, unexpected block number);
- timingViolation; and
- those that are vendor-defined.

12.15.2.4.9 EndUpload method

12.15.2.4.9.1 General

Table 253 describes the EndUpload method of the UploadDownload object.

Table 253 – UploadDownload object EndUpload method

Standard object type name: UploadDownload object				
Standard object type identifier: 3				
Method name	**Method ID (non-negative)**	**Method description**		
EndUpload	6	A client uses the EndUpload method to indicate that the upload operation is terminating		
	Input arguments			
	Argument number	**Argument name**	**Argument type**	**Argument description**
	1	Rationale	Unsigned8	This argument indicates the client's reason for terminating the upload operation
	Output arguments			
	Argument number	**Argument name**	**Argument type**	**Argument description**
	None			

12.15.2.4.9.2 Method description

A client uses the EndUpload method to indicate that the upload operation is terminating. Termination may occur for example if the upload has completed, or if the client has elected to terminate the upload operation.

EndUpload may be sent from a client that is presently engaged in an upload operation, as agreed by the StartUpload method.

12.15.2.4.9.3 Input arguments

The Rationale argument indicates the client's reason for terminating the upload operation. The value used shall be from the following set:

- 0: upload completed successfully; or
- 1: client abort.

12.15.2.4.9.4 Output arguments

There are no output arguments for this method.

12.15.2.4.9.5 Service feedback codes

The following feedback codes are valid for this method:

- success;
- operationIncomplete;
- unexpectedMethodSequence ;
- timingViolation;
- those that are vendor-defined.

12.15.2.4.10 State table for download

Table 254 shows the download state table.

Table 254 – Download state table for unicast operation mode *(1 of 2)*

Transition	Current State	Event(s)	Action(s)	Next State
T1	Idle	Execute.indicate(StartDownload)	Execute.response(success)	Downloading
T2	Downloading	Execute.indicate(DownloadData) Request for block is from same client object that started the download, and download data parameters are acceptable	Execute.response(success)	Downloading
		Execute.indicate(StartDownload) or Execute.indicate(any Upload Method)	Execute.response (objectStateConflict)	
		Execute.indicate(DownloadData) and request is from wrong client, or something is wrong with the download data parameters or timing	Execute.response(appropriate error) where the appropriate error may be, for example, invalidArgument, incompatibleMode, timingViolation, etc. NOTE It is a local matter for the UploadDownload object to determine if/when to abort the download.	
		Execute.indicate(EndDownload [Success]) and UploadDownload object does not agree download was completed successfully	Execute.response(incompatibleMode)	
		Write.indicate(StateCommand.Any value)	Write.response(objectStateConflict)	
T3	Downloading	Execute.indicate(EndDownload [Success])	Execute.response(Success)	DLComplete
T4	DLComplete	Write.indicate(StateCommand, Apply)	Write.response(operationAccepted)	Applying
T5	Applying	Application successful	None	Idle
T6	Downloading	Timeout waiting for subsequent method invocation	Update ErrorCode attribute of UploadDownload object	DLError
		Execute.indicate(EndDownload[Abort])	Update ErrorCode attribute of UploadDownload object. Execute.response (Success)	

Table 254 *(2 of 2)*

Transition	Current State	Event(s)	Action(s)	Next State
T7	Idle	Execute.indicate(any Download method other than StartDownload)	Execute.response(objectStateConflict)	Idle
		Execute.indicate(StartDownload) and Request is unacceptable. For example, one or more input arguments are not agreeable	Execute.response(appropriate error) (e.g., invalidObjectID)	
		Write.indicate(StateCommand.Any value other than Reset)	Write.response(objectStateConflict)	
		Write.indicate(StateCommand.Reset)	Write.response(success)	
T8	DLComplete	Execute.indicate(any Download method or any Upload method)	Execute.response(objectStateConflict)	DL_Complete
T9	Applying	Execute.indicate(any Download method or any Upload method)	Execute.response(objectStateConflict)	Applying
		Write.indicate(StateCommand.Any value)	Write.response(objectStateConflict)	
T10	DLComplete	Write(StateCommand, Reset)	1. Discard download content; and 2. Write.req(success)	Idle
T11	DLError	Write(StateCommand, Reset)	1. Discard download content; 2. Clear ErrorCode attribute; and 3. Write.req(success)	Idle
T12	DLError	Any Upload or Download method	Execute.response (objectStateConflict)	DLError
		Any state command other than Write(StateCommand.Reset)	Write.req(objectStateConflict)	
T13	Applying	Application failure	Update ErrorCode attribute of UploadDownload object	DLError
T14	DLComplete	Timeout waiting for command to apply	Update ErrorCode attribute of UploadDownload object	DLError

Figure 123 shows the UploadDownload object download state diagram.

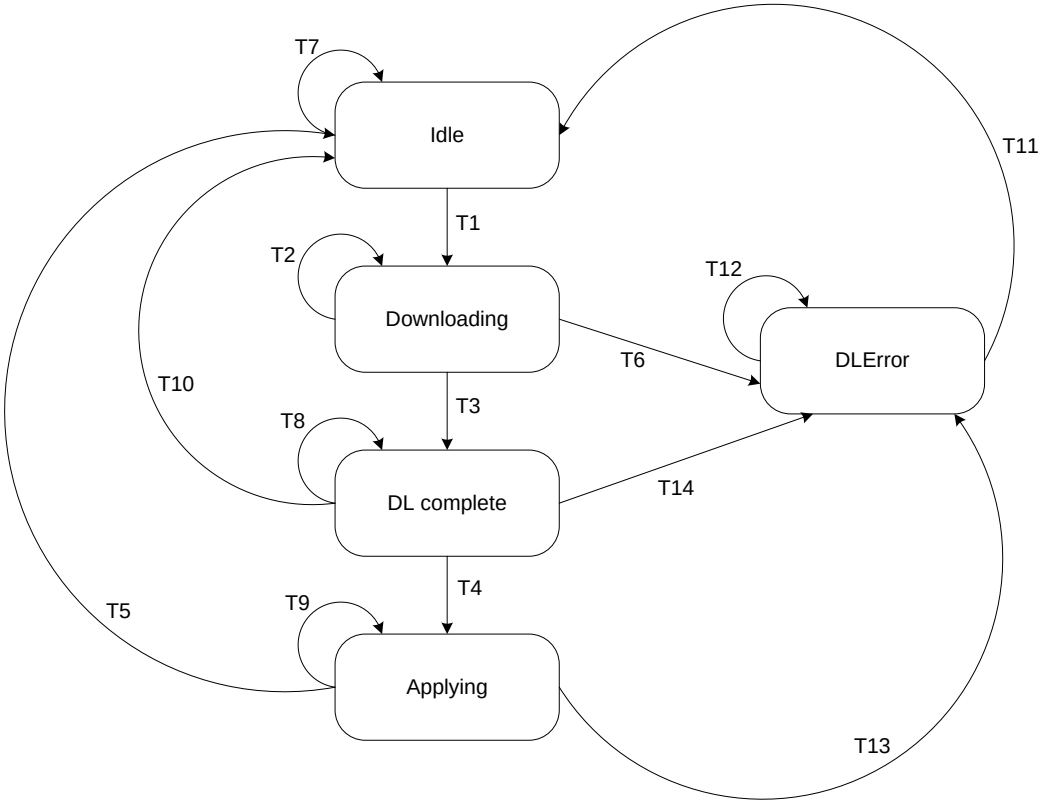


Figure 123 – UploadDownload object download state diagram

12.15.2.4.11 State table for upload

Table 255 shows the upload state table.

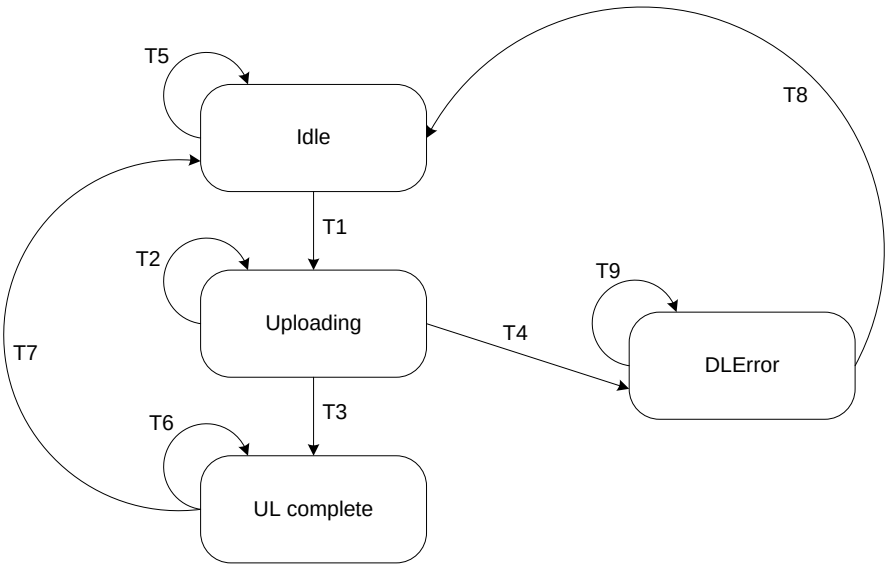


Figure 124 – UploadDownload object upload state diagram

Table 255 – Upload state table for unicast operation mode *(1 of 2)*

Transition	Current State	Event(s)	Action(s)	Next State
T1	Idle	Execute.indicate(StartUpload)	Execute.response(Success)	Uploading
T2	Uploading	Execute.indicate(UploadData) Request for block is from same client object that started the upload, and upload data parameters are as acceptable	Execute.response(Success)	Uploading
		Execute.indicate (StartUpload) or any Download method	Execute.response (objectStateConflict)	
		Execute.indicate(UploadData) and request is from wrong client, or something is wrong with the upload data parameters or timing	Execute.response(appropriate error) where the appropriate error may be, for example, invalidArgument, incompatibleMode, timingViolation, etc. NOTE It is a local matter for the UploadDownload object to determine if/when to abort the upload if this occurs more than once consecutively.	
		Execute.indicate(EndUpload [Success]) and UploadDownload object does not agree upload was successful	Execute.response(incompatibleMode)	
		Write.indicate(StateCommand .Any value)	Write.response(objectStateConflict)	
T3	Uploading	Execute.indicate(EndUpload [Success])	Execute.response(Success)	ULComplete
T4	Uploading	Timeout waiting for subsequent method invocation	Update ErrorCode attribute of UploadDownload object	UL_Error
		Execute.indicate(EndUpload [Abort])	1. Update ErrorCode attribute of UploadDownload object; 2. Execute.response(Success)	
T5	Idle	Execute.indicate(any Upload method other than StartUpload)	Execute.response(objectStateConflict)	Idle
		Execute.indicate(StartUpload) and request is unacceptable. For example, one or more input arguments are not agreeable	Execute.response(appropriate error) (e.g., invalidObjectID)	
		Write.indicate(StateCommand. any other than Reset)	Write.response(objectStateConflict)	
		Write.indicate(StateCommand.Reset)	Write.response(success)	

Table 255 *(2 of 2)*

Transition	Current State	Event(s)	Action(s)	Next State
T6	ULComplete	Execute.indicate(any DownloadMethod or any UploadMethod)	Execute.response (objectStateConflict)	UL_Complete
T7	ULComplete	Write(StateCommand, Reset)	Write.req(success)	Idle
T8	ULError	Write(StateCommand, Reset)	1. Clear ErrorCode attribute; and 2. Write.req(success)	Idle
T9	ULError	Any Upload or Download method	Execute.response (objectStateConflict)	ULError
		Write(StateCommand.other than Reset)	Write.req(error)	

Figure 124 shows the UploadDownload object's upload state diagram.

12.15.2.4.12 Client responsibilities for upload/download operations

In order to handle message delays in both requests and responses, and to avoid a congestion collapse due to a retransmission loop, only the first instance of a response indicating success shall cause the next data block to be sent via a DownloadData or requested via an UploadData method invocation by the client.

NOTE The intent is to avoid recreating historical situations such as occurred with the trivial file transfer protocol (TFTP), creating the Sorcerer's Apprentice Syndrome.

12.15.2.5 Concentrator object

12.15.2.5.1 General

A concentrator object represents an assembly of data, collected from multiple objects in the same UAP, that is to be published by a single publish request service. This object optimizes publication messages sent from a device. Multiple concentrator object instances may be used to represent multiple assemblies of data if required. A list of attributes is provided to indicate the data values that are published.

NOTE The published content represented by this object is established by configuration. This standard does not specify the device configuration tool.

A subscriber to data produced by a concentrator object shall only be a dispersion object. The data types associated with the list of attributes of the dispersion object should be configured to match those produced by the concentrator object.

When a concentrator object is configured by a host application, such as a gateway, the device is responsible for establishing contracts as needed to support the corresponding publications. The design is intended to support two use cases. In one case, the device joins the network and then the host configures the concentrator object. In the other case, the concentrator object is pre-configured and the device autonomously starts publication after it joins the network.

A UAP may have zero or more concentrator object instances.

12.15.2.5.2 Object attributes

A concentrator object has the attributes defined in Table 256.

The first time a UAP receives a read/write/execute request from an endpoint for which it has no contract, it shall request a contract so that it can send a service response to the requesting endpoint. The UAP shall, as necessary, delay the first service response to allow for time to establish/modify the contract.

Table 256 – Concentrator object attributes *(1 of 2)*

Standard object type name: Concentrator object				
Standard object type identifier: 4				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
ObjectIdentifier	Object key identifier	Unique identifier for the object	Type: Unsigned16 Classification: Constant Valid range: > 0	N/A
Reserved for future use	0	—	—	—
Concentrator ContentRevision	1	Tracks a change in what is published; ensures Concentrator (publisher) and Dispersion (subscriber) objects are in harmony	Type: Unsigned8 Classification: Static Accessibility: Read/write Default value: 0	Revision shall be incremented when the complement of data to publish changes, i.e. CommunicationEndpoint or Array of ObjectAttributeIndexAndSize is changed. Attribute included in Table 347 header
CommunicationEndpoint	2	Serves to identify the object that receives the publication from this object	Type: Communication association endpoint structure Classification: Static Accessibility: Read/write Default value: The configured connection endpoint valid element indicates not configured (i.e., endpoint is not valid)	Write to this attribute last when configuring this object; see Table 265
Communication contract data for scheduled communication	3	Data corresponding to the communication contract	Type: Communication contract data Classification: Static Accessibility: Read only	Updated when the corresponding contract is established or terminated; see Table 266
MaximumItems Publishable	4	Maximum number of items that can be published	Type: Unsigned8 Classification: Constant Accessibility: Read only Default value: Local matter	If this attribute has a value of 0, it indicates it is not configured for publishing
NumberItemsPublishing	5	Actual number of items being published	Type: Unsigned8 Classification: Static Accessibility: Read only Default value: 0	Updated as ObjectAttributeIndexAndSize attributes are configured : incremented when another value to publish is added, and decremented when a value to publish is removed

Table 256 *(2 of 2)*

Standard object type name: Concentrator object				
Standard object type identifier: 4				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
ObjectAttributes	6	Array of data to identify each piece of data published	Type: Array of Object AttributeIndexAndSize	Object ID, attribute ID, attribute index, and size for each value published. See Table 264
			Classification: Static	
			Accessibility: Read/write	
			Default value: Element size is 0	
Reserved for future use by this standard	7..63	—	—	—

Revision, NumItemsSubscribing, and ObjAttrIdx attributes can be implemented in a device in such a way that they can be written atomically in a single network transaction via concatenation of APDUs.

12.15.2.5.3 Standard object methods

A concentrator object has the methods defined in Table 257.

Table 257 – Concentrator object methods

Standard object type name: Concentrator object		
Standard object type identifier: 4		
Method name	**Method ID**	**Method description**
Null	0	Reserved by standard for future use
Reserved for future use by this standard	0..127	These method identifiers are reserved for future use by this standard
Implementation-specific use	128..255	These method identifiers are available for implementation-specific use

12.15.2.6 Dispersion object

12.15.2.6.1 General

A dispersion object is the subscribing object corresponding to a concentrator object. This object is configured to indicate how to parse a concentrator object's published content. If multiple disassemblies are required, multiple dispersion user objects are to be used. A UAP may have zero or more dispersion object instances.

NOTE Concentrator and dispersion objects are special objects supporting a publication proxy within a UAP. These objects are distinct from proxy application processes, which are able to distribute information across multiple UAPs within the UAL.

12.15.2.6.2 Object attributes

A dispersion object has the attributes defined in Table 258.

Table 258 – Dispersion object attributes *(1 of 2)*

Standard object type name: Dispersion object				
Standard object type identifier: 5				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
ObjectIdentifier	Object key identifier	Unique identifier for the object	Type: Unsigned16 Classification: Constant Valid range: > 0	N/A
Reserved for future use	0	—	—	—
Concentrator ContentRevision	1	Tracks changes to content subscribed. Ensures Concentrator publishing object and Dispersion subscribing object are in harmony	Type: Unsigned8 Classification: Static Accessibility: Read/write Default value: 0	Updated when the complement of data to publish changes. In the event of a mismatched ContentRevision (Table 347), the publication shall not be processed
CommunicationEndpoint	2	Endpoint of concentrator object that publishes data to this dispersion object	Type: Communication association endpoint structure Classification: Static Accessibility: Read/write Default value: The configured connection endpoint valid element indicates not configured (i.e., endpoint is not valid) Valid range: See structure definition	Write to this attribute last when configuring this object
MaximumItemsSubscribing	3	Maximum number of items that can be subscribed	Type: Unsigned8 Classification: Constant Accessibility: Read only Default value: Local matter Valid range: >0	Maximum number of items in corresponding publication
NumItemsSubscribing	4	Number of items being subscribed to	Type: Unsigned8 Classification: Static Accessibility: Read/write Default value: 0	Actual number of items in corresponding publication. A value of zero indicates the object is not configured to subscribe

Table 258 *(2 of 2)*

Standard object type name: Dispersion object				
Standard object type identifier: 5				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Array of ObjectAttributeIndexAndSize	5	Array of data to identify each piece of data published	Type: Array of ObjectAttributeIndexAndSize Classification: : Static Accessibility: Read/write Default value: Element size is 0	Object ID, Attribute ID, Attribute index, and size of data for the destination within the application for the published information NOTE To skip over data, the destination object and attribute may locally represent a Null object and Null attribute.
Reserved for future use by this standard	6..63	—	—	—

Revision, NumItemsSubscribing, and ObjAttrIdx attributes can be implemented in a device in such a way that they can be written atomically in a single network transaction via concatenation of APDUs.

12.15.2.6.3 Standard object methods

A dispersion object has the methods defined in Table 259.

Table 259 – Dispersion object methods

Standard object type name: Dispersion object		
Standard object type identifier: 5		
Method name	**Method ID**	**Method description**
Null	0	Reserved by standard for future use
Reserved for future use by this standard	0..127	These method identifiers are reserved for future use by this standard
Implementation-specific use	128..255	These method identifiers are available for implementation-specific use

12.15.2.7 Tunnel object

12.15.2.7.1 General

The tunnel object (TUN) is used to support the energy efficient transport of encapsulated messages over the network for a single non-native protocol. The tunnel service and a variation of the publication service are defined for this encapsulation. Support structures are provided for deconstruction, mapping and reconstruction of non-native protocol packets in order to reduce transactions and packet size.

NOTE The usage of the tunnel object to create protocol translators is intended to be defined by the organization that has defined the non-native protocol used in the tunnel.

12.15.2.7.2 Object attributes

A tunnel object has the attributes defined in Table 260.

Table 260 – Tunnel object attributes *(1 of 3)*

Standard object type name: Tunnel object				
Standard object type identifier: 6				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
ObjectIdentifier	Object key identifier	Unique identifier for the object	Type: Unsigned16 Classification: Constant Valid range: > 0	N/A
Reserved for future use	0	—	—	—
Protocol	1	Type of protocol supported by this object	Type: Unsigned8 Classification: Constant Accessibility: Read only Default value: Local matter (protocol-specific) Valid range: See Annex M	Sets the specific protocol that is encapsulated in tunnel messages. Only matching protocol tunnels exchange meaningful data
Status (configuration status)	2	Communication configuration status of this object	Type: Unsigned8 Classification: Static Accessibility: Read/write Default value: 0 Named values: 0: not configured; 1: validly configured; 2: invalidly configured	The object status is not configured when the Protocol is set to None and no communication occurs. Once the object is configured and another protocol is set, the object attempts to apply the configuration and changes the status appropriately
Flow_Type	3	Communication service used by this object	Type: Unsigned8 Classification: Static Accessibility: Read/write Named values: 0: 2-part tunnel; 1: 4-part tunnel; 2: publish; 3: subscribe	Configures the tunnel for a specific type of communication and role
Update_Policy	4	Periodic communication update policy for this object	Type: Unsigned8 Classification: Static Accessibility: Read/write Named values: 0: periodic; 1: change of state	Sets the periodic publication policy for a linked publisher and subscriber. A periodic update publishes on every opportunity. Change of state publishes on fresh data or at least as often as Stale_Limit specifies

Table 260 *(2 of 3)*

Standard object type name: Tunnel object				
Standard object type identifier: 6				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Period (data publication period)	5	Periodic communication update period for this object	Type: Integer16 Classification: Static Accessibility: Read/write Default value: 0 Named values: 0: not configured	Sets the periodic publication time for a linked publisher and subscriber. Isochronous publication is enabled by an implicit rule. Publication does not begin until a period is set. See 12.12.5
Phase (ideal publication phase)	6	Periodic communication phase within period for this object	Type: Unsigned8 Classification: Static Accessibility: Read/write Valid range: 0..99	Sets the requested publication time within the period for a linked publisher and subscriber. The actual phase may differ by contract requirements. Units should be indicated as a percentage (%)
Stale_Limit (stale data limit)	7	Periodic communication stale data limit for this object	Type: Unsigned8 Classification: Static Accessibility: Read/write	Defines the maximum subscriber expected arrival time as a multiple of the period. Defines the minimum publication rate for change of state reporting as a multiple of the period
Max_Peer_Tunnels	8	Maximum number of correspondent tunnels with which this object can communicate	Type: Unsigned8 Classification: Constant Accessibility: Read only	N/A
Num_Peer_Tunnels	9	Actual number of correspondent tunnels with which this object is communicating	Type: Unsigned8 Classification: Static Accessibility: Read/write Default value: 0	Incremented / decremented as Tunnel endpoints array elements are added and deleted
Array of Tunnel endpoint	10	Array of Protocol association endpoints	Type: Array of Tunnel endpoint Classification: Static Accessibility: Read/write Valid range: Address information pointing to one or more tunnel objects	Links remote tunnel objects for communication with this tunnel object

Table 260 *(3 of 3)*

Standard object type name: Tunnel object				
Standard object type identifier: 6				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Foreign_Source_Address	11	Foreign source address mapped to this objects communication	Type: IPv6Address Classification: Static Accessibility: Read/write	Holds static addressing information to be delivered to initiator or correspondent upon message receipt
Foreign_Destination_Address	12	Foreign destination address mapped to this object's communication	Type: IPv6Address Classification: Static Accessibility: Read/write	Holds static addressing information to be delivered to initiator or correspondent upon message receipt
Connection_Info[ ]	13	Foreign connection information mapped to this object's communication	Type: OctetString Classification: Static Accessibility: Read/write	Holds static information to be delivered to initiator or correspondent upon message receipt
Transaction_Info[ ]	14	Foreign transaction information mapped to this object's communication	Type: OctetString Classification: Dynamic Accessibility: Read/write	Holds transaction specific information to be delivered to initiator on completion of a transaction
Reserved for future use by this standard	15..63	—	—	

12.15.2.7.3 Standard object methods

A tunnel object has the methods defined in Table 261.

Table 261 – Tunnel object methods

Standard object type name: Tunnel object		
Standard object type identifier: 6		
Method name	**Method ID**	**Method description**
Null	0	Reserved by standard for future use
Reserved for future use by this standard	0..127	These method identifiers are reserved for future use by this standard
Implementation-specific use	128..255	These method identifiers are available for implementation-specific use

12.15.2.8 Interface object

12.15.2.8.1 General

The interface object provides a generic messaging end point for interfacing to a network. This object may be used as the source or destination object in native messaging interactions required for support of gateway protocol translation and various native messaging applications.

The interface object may be indicated in client/server communication services necessary for native read, write, and execute services. The interface object may also be referenced as the client object communicating with an upload/download object for bulk transfer.

Communications referencing the interface object as a client shall adhere to the client/server congestion control policies defined in this standard.

Where possible, implementers shall consider buffering client server retrieved values for local usage rather than creating additional communications over the wireless network to repeatedly retrieve these values from devices which need to preserve power. User application objects contained in the field devices provide guidance, via their specification of attribute data classification, regarding what object-related information should be buffered.

NOTE 1 Native object publication and subscription are accomplished by using the concentrator and dispersion objects.

NOTE 2 The actual structure of buffering/cache and local requirements for handling messages to the cache are considered local matters, outside the scope of this standard.

12.15.2.8.2 Object attributes

An interface object has the attributes defined in Table 262.

Table 262 – Interface object attributes

Standard object type name: Interface object				
Standard object type identifier: 7				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
ObjectIdentifier	Object key identifier	Unique identifier for the object	Type: Unsigned16 Classification: Constant Valid range: > 0	N/A
Reserved for future use	0	—	—	—
Reserved for future use by this standard	1..63	—	—	—

12.15.2.8.3 Standard object methods

An interface object has the methods defined in Table 263.

Table 263 – Interface object methods

Standard object type name: Interface object		
Standard object type identifier: 7		
Method name	**Method ID**	**Method description**
Null	0	Reserved by standard for future use
Reserved for future use by this standard	0..127	These method identifiers are reserved for future use by this standard
Implementation-specific use	128..255	These method identifiers are available for implementation-specific use

12.16 Data types

12.16.1 Basic data types

The basic data types supported for attributes are:

- binary values;
- 8-, 16-, and 32-bit signed integers;
- 8-, 16-, 32-, 64- and 128-bit unsigned integers;
- ISO/IEC/IEEE 60559 (IEEE 754) 32-bit and 64-bit floating point values;
- strings representing visible text, a block of octets, or a sequence of bit values (BitString);
- time: TAINetworkTime, TAITimeDifference, TAITimeRounded.

12.16.2 Derived atomic data types

The derived atomic data types supported for attributes are:

- addresses:
  - IPv6Address (mapped to a 128-bit unsigned integer),
  - EUI64Address (mapped to a 64-bit unsigned integer),
  - DL16Address (mapped to a 16-bit unsigned integer);
- layer-specific identifiers (generally mapped to 8-bit or 16-bit unsigned integers);
- MIB indices (generally mapped to 8-bit or 16-bit unsigned integers).

12.16.3 Industry-independent standard data structures

12.16.3.1 General

Standard data structures used shall be the data structures conveyed by the protocol defined by this standard. Industry-independent standard data structures are summarized in Annex L.

NOTE Vendor-specific data structure definitions are not supported.

12.16.3.2 Object, attribute, index, and size

The elements of ObjectAttributeIndexAndSize are shown in Table 264.

Table 264 – Data type: ObjectAttributeIndexAndSize

Standard data type name: ObjectAttributeIndexAndSize		
Standard data type code: 469		
Element name	**Element identifier**	**Element scalar type**
ObjectID	1	Type: Unsigned16 Classification: Static Accessibility: Varies by use
AttributeID	2	Type: Unsigned16 Classification: Static Accessibility: Varies by use
AttributeIndex	3	Type: Unsigned16 Classification: Static Accessibility: Varies by use
Size	4	Type: Unsigned16 Classification: Static Accessibility: Varies by use NOTE In practice, this maximum size depends on the capabilities of the device.

12.16.3.3 Communication association endpoint

The data structure shown in Table 265 is used for communication endpoints for both inputs and outputs.

Table 265 – Data type: Communication association endpoint *(1 of 2)*

Standard data type name: Communication association endpoint		
Standard data type code: 468		
Element name	**Element identifier**	**Element scalar type**
Network address of remote endpoint	1	Type: IPv6Address This is a logical construct configured for the device by the system manager NOTE The system manager ensures that such configuration supports both device replacement and mobile device scenarios. Classification : Static Accessibility: Read/write
T-port at remote endpoint	2	Type: Unsigned16 Classification : Static Accessibility: Read/write
Object ID at remote endpoint	3	Type: Unsigned16 Classification : Static Accessibility: Read/write
Stale data limit	4	Type: Unsigned8 Classification : Static Accessibility: Read/write NOTE 1 This attribute is primarily of interest to a subscriber. NOTE 2 This is a count of consecutive stale input values that a subscriber fails to receive before the subscriber considers the value previously received to be Bad (Table 299). Staleness is implied by an unchanging freshness sequence number (Table 347).
Data publication period	5	Type: Integer16 Classification : Static Accessibility: Read/write NOTE For units of time, see 12.12.5.
Ideal publication phase	6	Type: Unsigned8 Classification : Static Accessibility: Read/write Valid range: 0..99 (as a percentage %) NOTE This attribute is primarily of interest to a publisher.
PublishAutoRetransmit	7	Type: Unsigned1 Classification: Static Accessibility: Read/write Named values[a] 0: transmit only if application content changed since last publication; 1: transmit at every periodic opportunity (regardless of whether application content changed since last transmission or not)

Table 265 *(2 of 2)*

Standard data type name: Communication association endpoint		
Standard data type code: 468		
Element name	**Element identifier**	**Element scalar type**
Configuration status	8	Unsigned8 Classification: Static Accessibility: Read access Named values: 0 : not configured (connection endpoint not valid); 1: configured (connection endpoint valid) NOTE The data owner sets this element to a value of 0 to indicate that the endpoint is not configured, and to 1 to indicate the endpoint is configured. An endpoint is considered not configured if the value of Object ID at remote endpoint is 0.
a The coding of this attribute is the inverse of the related attribute 12 of Table 27.		

12.16.3.4 Communication contract data

The data structure shown in Table 266 is used for the dynamic data important to the local application process that is associated with a particular communication contract.

NOTE 1 It is a local matter to ensure that applications are well-behaved in terms of the communication contracts they employ. The AL does not standardize the policing of compliance with requested contracts.

NOTE 2 As part of contract negotiation, sufficient information is provided to the contract requesting device in order to enable it to determine the maximum size APDU that the contract supports. For example, if contract negotiation determines the maximum network service data unit (NSDU) size, then the type of security in effect for the contract is locally determinable for the contract. If the type of security in use is known, the transport header size is locally acquired and subtracted from the maximum NPDU size, thus yielding the value for the maximum APDU size usable for communications employing that particular communication contract.

Table 266 – Data type: Communication contract data

Standard data type name: Communication contract data		
Standard data type code: 470		
Element name	**Element identifier**	**Element type**
ContractID	1	Type: Unsigned16 Classification: Static Accessibility: Read only Valid range: The set of valid values is defined by the system management
Contract_Status	2	Type: Unsigned8: Classification: Static Accessibility: Read only Default value = 0 Named values: 0: endpoint_not_configured, 1: awaiting_contract_establishment, 2: contract_active_as_requested, 3: contract_active_negotiated_down, 4: awaiting_contract_termination, 5: contract_establishment_failed, 6: contract_inactive
Actual_Phase	3	Type:Unsigned8 Classification: Dynamic Accessibility: Read only Default value: 0 (indicating not assigned) Valid range: 0..99 (in units of percentage %)
Further information on the actual contract, such as the negotiated-down parameters, may be available from the DMAP and does not need to be maintained by the UAP.		

12.16.3.5 Alert communication endpoint

The data structure shown in Table 267 is used for communication endpoints for alert reports.

Table 267 – Data type: Alert communication endpoint

Standard data type name: Alert communication endpoint		
Standard data type code: 471		
Element name	**Element identifier**	**Element scalar type**
Network address of remote endpoint	1	Type: IPv6Address This is a logical construct configured for the device by the system manager NOTE The system manager ensures that such configuration supports both device replacement and mobile device scenarios. Classification : Static Accessibility: Read/write
T-port at remote endpoint	2	Type: Unsigned16 Classification : Static Accessibility: Read/write
Object ID at remote endpoint	3	Type: Unsigned16 Classification : Static Accessibility: Read/write

12.16.3.6 Tunnel endpoint

The data structure shown in Table 268 is used in tunnel objects to identify remote tunnel endpoints for exchange of encapsulated payloads.

Table 268 – Data type: Tunnel endpoint

Standard data type name: Tunnel endpoint		
Standard data type code: 475		
Element name	**Element identifier**	**Element scalar type**
Network_Address (network address of remote endpoint)	1	Type: IPv6Address This is a logical construct configured for the device by the system manager NOTE The system manager ensures that such configuration supports both device replacement and mobile device scenarios. Classification: Static Accessibility: Read/write
Transport_Port (T-port at remote endpoint)	2	Type: Unsigned16 Classification: Static Accessibility: Read/write
OID (object ID at remote endpoint)	3	Type: Unsigned16 Classification: Static Accessibility: Read/write

12.16.3.7 Alert report descriptor

Elements of the alert report descriptor are shown in Table 269.

Table 269 – Data type: Alert report descriptor

Standard data type name: Alert report descriptor		
Standard data type code: 499		
Element name	**Element identifier**	**Element type**
Alert report disabled	1	Type: Boolean8 Classification: Static Accessibility: Read/write Default value : Local matter
Alert report priority	2	Type: Unsigned8 Classification: Static Accessibility: Read/write Default value : 0 Valid range: 0..15, as specified in 12.17.5.2.2.22

12.16.3.8 Analog alarm descriptor

The analog alarm descriptor is used to define alarm reporting for an analog value with a single reference condition. Its elements are shown in Table 270.

Table 270 – Data type: Process control alarm report descriptor for analog with single reference condition

Standard data type name: Process control alarm report descriptor for analog with single reference condition		
Standard data type code: 498		
Element name	**Element identifier**	**Element scalar type**
Alert report disabled	1	Type: Boolean8 Classification: Static Accessibility: Read/write Default value : TRUE
Alert report priority	2	Type: Unsigned8 Classification: Static Accessibility: Read/write Default value : 0 Valid range: 0..15
Alarm limit	3	Type: Float32 Classification: Static Accessibility: Read/write

12.16.3.9 Binary alarm descriptor

The binary alarm descriptor is the same structure as the data type for the alert report descriptor, so no additional data type description is required.

12.16.3.10 ObjectIDandType

The elements of ObjectIDandType are shown in Table 271.

Table 271 – Data type: ObjectIDandType

Standard data type name: ObjectIDandType		
Standard data type code: 472		
Element name	**Element identifier**	**Element scalar type**
ObjectID	1	Unsigned16
ObjectType	2	Unsigned8
ObjectSubType	3	Unsigned8
VendorSubType	4	Unsigned8

12.16.3.11 Unscheduled correspondent

The elements of UnscheduledCorrespondent are shown in Table 272.

Table 272 – Data type: UnscheduledCorrespondent

Standard data type name: UnscheduledCorrespondent		
Standard data type code: 473		
Element name	**Element identifier**	**Element scalar type**
Address	1	IPv6Address
T-port	2	Unsigned16

12.17 Application services provided by application sublayer

12.17.1 General

All interfaces between the DLE and adjacent layer (or sublayer) entities or management entities are internal interfaces within the device, and thus are unobservable. Therefore they are not subject to standardization.

Application services are provided by the ASL (at the ASLDE-n SAP) for communication with native objects, which are either UAP objects or MP objects. These are the only services that should be used for behavior compliant with this standard. Not all devices will need to use all of the services defined herein, and not all objects will support all the services herein. However, if these services are employed for communication between or among native objects, they should be employed as defined in this standard.

Application processes using ASL services should be designed to tolerate receipt of duplicate ASL service indications and confirmations. For example, if a lower layer acknowledgment is lost when a response to a read request is sent, the lower layer may retry, and as a result the application client may receive a duplicate response to the read request.

It is left to the device to determine how best to handle congestion/back pressuring if locally indicated by the local lower protocol suite. This handling may, for example, limit transmission of messages from the device for a certain period of time, or for a certain set of communication priorities, or both. Congestion may occur, for example, in situations of network communication load, and handling by the device is intended to limit additional congestion.

NOTE 1 Capacity planning is a systems issue and is outside the scope of AL consideration.

Table 273 summarizes the services provided.

NOTE 2 Local services that do not result in network communication are not included in Table 273, as they are local matters and hence implementation-dependent.

NOTE 3 Local ASL service confirmation back to the AP is a local matter, and hence is not defined by this standard.

Table 273 – AL services

ASL-provided service	Applicable primitives	Description	How used
Object access services			
Read	Request Indication Response Confirmation	Read an attribute value from an object	Client/server
Write	Request Indication Response Confirmation	Write a value to an object	Client/server
Execute	Request Indication Response Confirmation	Execute a method on an object	Client/server
Publication services			
Publish	Request Indication	Publish a single or multiple values from one source object	Publish/subscribe NOTE Native content, as well as non-native content, is supported.
Alert report-related services			
AlertReport	Request Indication	Report an alert	Source/sink (unicast only). The source of this service shall only be the ARMO. The sink of this service shall only be the alarm receiving object
AlertAcknowledge	Request Indication Response Confirmation	Acknowledge an individual alert reception	Client/server. The source of this service may only be an alarm receiving object
Explicit support for tunneling			
Tunnel	Request Indication Response Confirmation	Tunnel payload without ASL parsing (for non-native protocol compatibility)	Tunnel payload without ASL parsing (for non-native protocol compatibility). This service shall be used only if the source and destination objects are both tunnel objects

NOTE 4 If a local service request is malformed, reporting the error to the requesting UAP is a local matter, and thus is not addressed in this standard. If desired, it is possible for an implementation to keep statistics regarding locally received services requests that are malformed.

12.17.2 Publish/subscribe application communication model

Publication is a communication process that is initiated by an object in the publishing UAL and received by an object in the subscribing UAL. Publication uses ASL services specific to the supporting publication. Publication occurs from a publisher object to a subscribing object. Any object may act as publisher or subscriber. To optimize communication bandwidth usage, special objects (a concentrator object for publishing and a dispersion object for subscribing) are defined to enable publication from/to a set of objects within a single UAP using a single publish service invocation.

The semantic reason for publishing is purely an application concern. This model supports communication for:

- schedule-triggered periodic buffered communications;
- application-triggered buffered communications; and
- change-of-state-triggered buffered communications.

All of the above published communications use the publish/subscribe communication flow paradigm. For scheduled periodic communications, subscriber applications may support timeout response methods to deal with a loss of individual publications and/or a loss of the publisher endpoint. For example, a subscribing application may use prior publication value content, but may degrade the corresponding quality of the value. Loss of individual messages may have a shorter timeout than the timeout used by the subscriber to determine the loss of the publisher.

Coordination of a publication with the network schedule is accomplished via appropriate endpoint configuration, which drives communication contract requests. A communication contract request is used to request that the system manager allocate scheduled bandwidth for publish communication.

NOTE 1 Determination of actual timeout policy when an expected publication is not received is an application process-specific matter that is not specified by this standard.

NOTE 2 Published messages with native content always contain a sequence counter and the current data value(s). If there is no change in value, the sequence counter indicates that the publishing application is still operating and has no new data to report (this is also known as a heartbeat). Some receiving devices retain this sequence counter to determine if the value has changed in order to limit reprocessing, while other receiving devices elect to ignore the sequence counter.

NOTE 3 For scheduled periodic communications, application processes often use common network time to synchronize their activities across the network. This synchronization is locally applicable by publishers and subscribers to synchronize their activities to the publication schedule.

NOTE 4 Continuous data and measurement in this standard employ a control system field proven publish/subscribe communication model and leverage use of scheduled bandwidth for more precise communication timing. If there is a more customized communication requirement, it is considered a custom situation outside the scope of this standard, or possibly a situation for future consideration.

12.17.3 Scheduled periodic buffered communication

12.17.3.1 General

The publish service is used for unidirectional buffered communication to at most one subscriber.

Publishing should be configured in accordance with the capabilities of the system. If a subscriber is not present, it is generally an interim or error situation and is not expected to be long-lived. Potential energy loss due to improperly configured or failed devices may be addressed by reconfiguration or device replacement. These are rare abnormal situations for which design optimization is not required; hence, the complexity required for publication only in the presence of an actual subscriber outweighs any communications savings.

No acknowledgments or retries are applied to publish/subscribe interactions. A publishing ASL is given an unconfirmed service request and constructs an appropriate APDU, which is then passed to the lower communication layers for communication transfer. If a publishing ASL receives a request for service before the prior request has been conveyed, the new request should overwrite the previous request and the previous request should not be transmitted. If a subscribing ASL receives a new request before the previous request was delivered to the destination object, the new request overwrites the previous request, and the previous request is lost.

The defined services support publication of an arbitrary attribute (for example, a process variable) from a simple device, or publication of a set of attributes from a more complex (for

example, a multi-variable capable) wireless device. Publish/subscribe communications take place over a configured communication relationship, as shown in Figure 125.

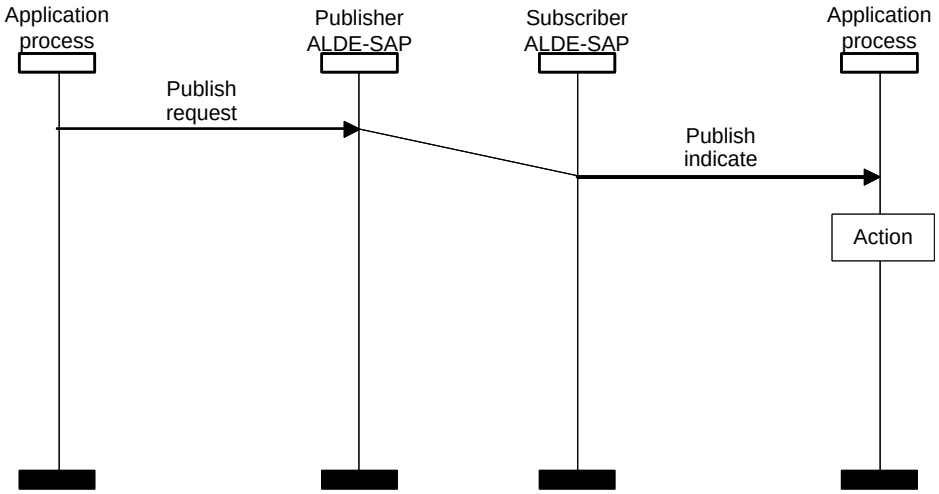


Figure 125 – Publish sequence of service primitives

The subscriber for a publication may be, for example, a gateway.

NOTE 1 The content of non-native publications is outside the scope of this standard.

Native publications include the attribute value itself, and status information for the value. In order to support duplicate detection and out-of-order delivery, a simple one-octet monotonic counter is included with each published value.

NOTE 2 Other schemes for uniquely identifying a published message, such as using a timestamp instead of a counter, were considered but eliminated because they incur more power to communicate. Time of TPDU construction, combined with a lower layer-provided freshness indication, often is locally available. Timestamps on data publications are useful in remote terminal unit (RTU)-style buffering gateways, but for such gateways, a reception timestamp is also usable for that purpose, particularly since all publications use a shared channel-hopping schedule, leaving a small variance between data generation time and data reception time.

Publish/subscribe message priority may be fixed; in that case, a local implementation may elect not to provide per-message priority. If an application requires publications with different priority levels, such as low priority publication for control monitoring and high priority publication for high-rate (1 Hz and 4 Hz) control loops variables, separate publish/subscribe relationships may be required.

The AL publisher and AL subscriber do not communicate explicitly to establish or break their relationship; however, establishment of secure communication relationships may force transport relationships to be established. Publishers and subscribers each establish their portion of the relationship independently (asynchronously). That is, either the publishing UAP or any of the subscribing UAPs may act first to establish its part of a publish/subscribe relationship. End-to-end messaging cannot commence before both the publisher and the subscriber(s) have established their respective sides of the communication relationship. Once the publisher creates one side of the communication relationship and a subscriber creates the other side of the communication relationship, publication messages can be sent and delivered to the corresponding application objects.

Locating publishers dynamically using either a tag discovery service or a centralized directory lookup service is outside the scope of this standard. Therefore, publish/subscribe is intended to be compatible with a static configuration mechanism. In future releases, a discovery mechanism may be employed. In either situation, the same information needs to be available to establish the publish/subscribe relationship.

Communication routes are formed transparently to the AL.

NOTE 3 The formation of transmission routes in support of publish/subscribe is important to ensure timely delivery, but is left as a responsibility for the system manager, which forms routes to use during communication contract establishment. See 6.3.11 for further details.

Publication is always an unconfirmed data transfer service request. Subscription always results in receiving an unconfirmed data transfer service indication.

NOTE 4 It is considered a management topic to ensure security configuration/changes support publish/subscribe without AL impact. It is understood that security considerations often constrain permitted relationships.

The timing of a scheduled publication is coordinated across the network, and as such depends on a coordinated view of time across the network. Bandwidth allocated for schedule-triggered publications needs to be reserved to ensure that subscribers can receive what their publishers send. The schedule should be configured to ensure best effort to meet delivery deadlines, but ultimately, the responsibility of the publisher to create new publications, and the subscriber to act on receipt of them, depends on the device's internal scheduling of the application process.

Published communications rely on the publication service support provided by the communication protocol suite that is underlying the ASL. An important aspect of this lower communication protocol suite is the ability to provide specific communication timing in order to meet scheduling demands.

12.17.3.2 Publish

12.17.3.2.1 General

The publish service for this standard is a unicast service used to update data periodically from a single publication source in a single AP to (at most) a single subscriber destination.

The publish service may also be used in an aperiodic manner to support both application-triggered and change-of-state-triggered changes. Since buffer content is transmitted according to a schedule, native published communication includes a freshness indicator to enable the subscriber to determine whether or not a value has changed.

NOTE Freshness does not mean unchanged data, but rather that the value has been newly (freshly) acquired since it was last published.

Table 274 defines the service primitives.

Table 274 – Publish service

Parameter name	Request	Indication
Argument	M	M
Service contract identifier	M	—
Priority	M	—
Discard eligible	M	—
End-to-end transmission time	—	M
Published data size	M	M
Subscriber T-port	M	—
Subscriber TDSAP	—	M
Subscribing object identifier	M	M(=)
Publisher IPv6Address	—	M
Publisher TDSAP	M	—
Publisher T-port	—	M(=)
Publishing object identifier	M	M(=)
DataStructureInformation	M	M(=)
NativeIndividualValue	S	S(=)
Freshness sequence number	M	M(=)
Individual analog value and status	S	S(=)
Individual digital value and status	S	S(=)
NativeValueList	S	S(=)
Publishing content version	M	M(=)
List of publish data	M	M(=)
Fresh value sequence number	M	M(=)
Analog value and status	S	S(=)
Digital value and status	S	S(=)
Non-native	S	S(=)
Non-native data	M	M(=)

12.17.3.2.2 Arguments

12.17.3.2.2.1 Service contract identifier

This parameter identifies the communication service contract agreement that was made between the UAP requesting the service and the local DMAP. The value shall be in the set of valid values for a contract identifier as defined by 6.3.11.

12.17.3.2.2.2 Priority

This parameter defines the message priority of service that is required of the communication. The permitted values for this service parameter may be an indication of either a high priority message or a low priority message. Transmission and delivery of high-priority messages is more important than transmission and delivery of messages of low priority.

12.17.3.2.2.3 Discard eligible

This parameter defines the guidance to the communication network regarding the application impact of discarding the application message in the event of network congestion. Possible values are TRUE (the message may be considered for discard), or FALSE (do not consider the message for discard).

NOTE This guidance is provided for use by routers that are constructed to employ an intelligent message discard policy rather than a random discard policy in situations of network congestion.

12.17.3.2.2.4 End-to-end transmission time

This is the transmission time from the TLE at the requesting device to the TLE at the receiving device. The interval is marked by two instants, the first instant being delivery to the TLE in the requesting device, and the second instant being receipt by the TLE in the destination device.

12.17.3.2.2.5 Published data size

This parameter provides the subscriber with the number of octets of the data to publish.

12.17.3.2.2.6 Subscriber T-port

This parameter identifies the subscriber UAP's associated T-port.

12.17.3.2.2.7 Subscriber TDSAP

This parameter identifies the subscriber TDSAP associated with the subscriber T-port.

12.17.3.2.2.8 Subscribing object identifier

This parameter specifies the object identifier destination in the application that is subscribed to this publication.

12.17.3.2.2.9 Publisher IPv6Address

This identifies the IPv6Address of the publisher.

12.17.3.2.2.10 Publisher TDSAP

This parameter uniquely identifies the publisher UAPs associated with the TDSAP. The TDSAP maps 1-to-1 to a UAP. The value shall be a member of the set of valid TDSAPs, as specified by the TL.

12.17.3.2.2.11 Publisher T-port

This parameter identifies the publisher UAP's associated T-port.

NOTE An implementation is able to infer this parameter from the publisher TDSAP. Thus it is included here for completeness, as required by this standard for the logical mapping to the transport data service request definition.

12.17.3.2.2.12 Publisher object identifier

This parameter identifies the publisher object that is the source of the published data.

NOTE If there is more than one entry in the list of published data, the publishing object source is an instance of the concentrator object type.

12.17.3.2.2.13 DataStructureInformation

This parameter indicates the construct of the information to be conveyed via publication. It may indicate one of the following constructs:

- native individual value;
- native sequence of values; or
- non-native data (that is, information being tunneled via a publication service).

The details of these alternatives are as follows:

a) The data structure of each single native value is as follows:

1) Freshness value sequence number

This parameter is present if the data structure information indicates the structure of the data is a native individual value. This parameter indicates the freshness of the data.

2) Individual analog value and status

This parameter is present if the individual native value is an analog. This contains standard value status data structure that indicates information such as quality of the corresponding analog value and the analog value itself.

3) Individual digital value and status

This parameter is present if the individual native value is digital. This contains standard value status data structure that indicates information such as quality of the corresponding digital value and the digital value itself.

b) The data structure of a list of native values is as follows:

1) Publishing content version

This parameter is present if the publication is for native list data, such as sent from a concentrator object. This information ensures harmonious interpretation of the published information by the subscriber.

2) List of publish data

This parameter represents the list of data conveyed via the publish service.

3) Status and analog value

This contains standard value status data structure that indicates information such as quality of the corresponding analog value and the analog value itself.

4) Status and digital value

This contains standard value status data structure that indicates information such as quality of the corresponding digital value and the digital value itself.

c) The data contained in the publish service that is non-native is as follows:

This parameter contains the non-native data to publish. Non-native data is conveyed as a string of octets.

12.17.4 Client/server interactions

12.17.4.1 General

Client/server interactions are used for one-to-one aperiodic communications. These relationships employ on-demand queued bidirectional communication. Client/server services defined by this standard are either two-part service (having two service primitives, .req and .ind), as in Figure 126, or four-part service (having four service primitives, .req, .ind, .rsp, and .cnf), as in Figure 127, Figure 128 and Figure 129.

When the ASL receives a client/server service request, it constructs a corresponding application protocol data unit (APDU) and requests queued transfer from the lower communication protocol suite. The server ASL is given a confirmed service response indication, which it delivers to the destination UAP and object.

For services with four-part primitives defined, the server UAP constructs a corresponding response. When the ASL receives the client/server service response, it constructs a response APDU and submits it to its lower communication protocol suite, which provides queue-oriented communication services to deliver the response.

In a client/server interaction, either endpoint of the communication can act as client or server or both. A client request is sent to a single destination (server). This request indicates the destination to which the response should be sent. A single response is then issued from the server.

Interactions as shown in Figure 127, Figure 128, and Figure 129 require a communication contract identifier and the communication protocol suite to be appropriately configured to support the client/server messaging requirements of the application. The bandwidth represented by the contract identifier is considered unscheduled shared bandwidth which need not be reserved solely for use by this contract.

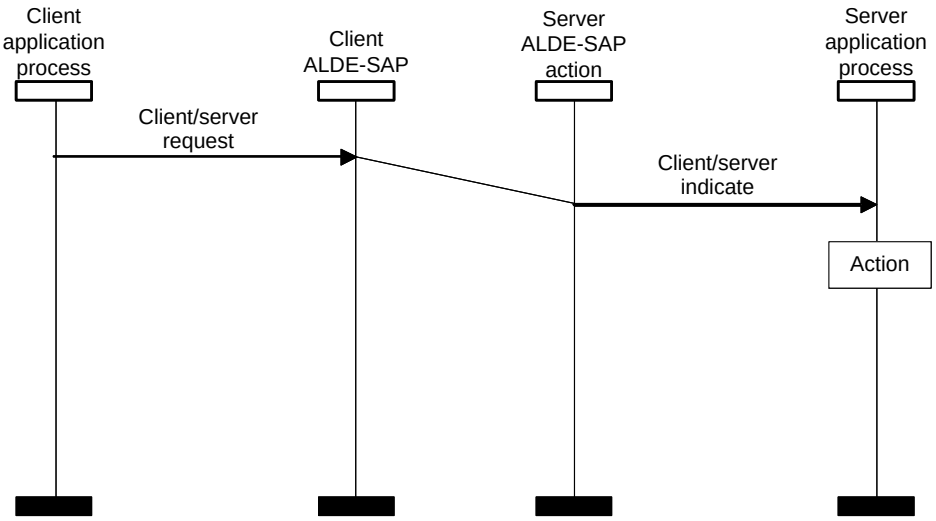


Figure 126 – Client/server model two-part interactions

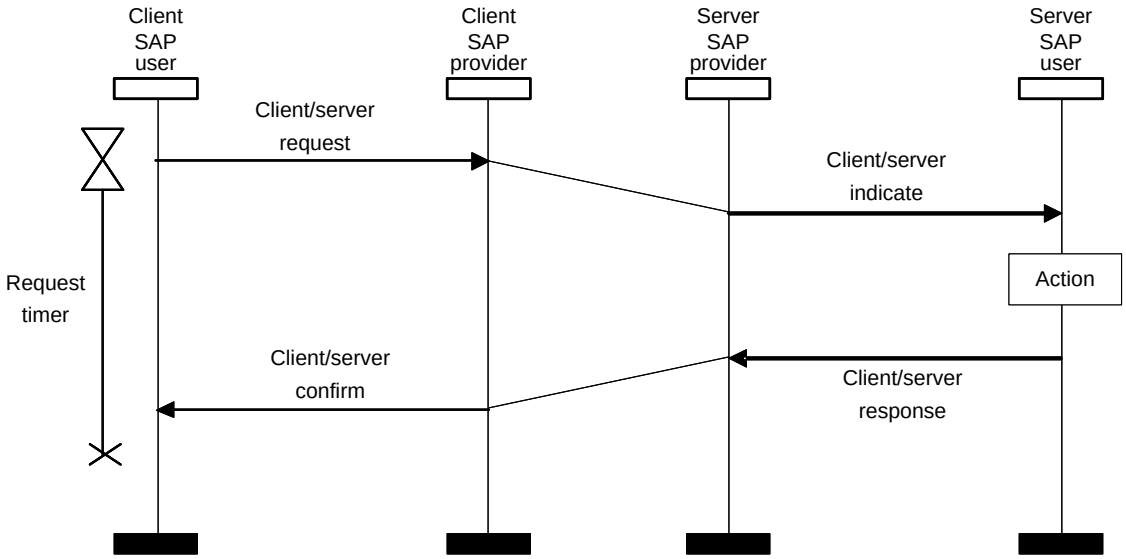


Figure 127 – Client/server model four-part interactions: Successful delivery

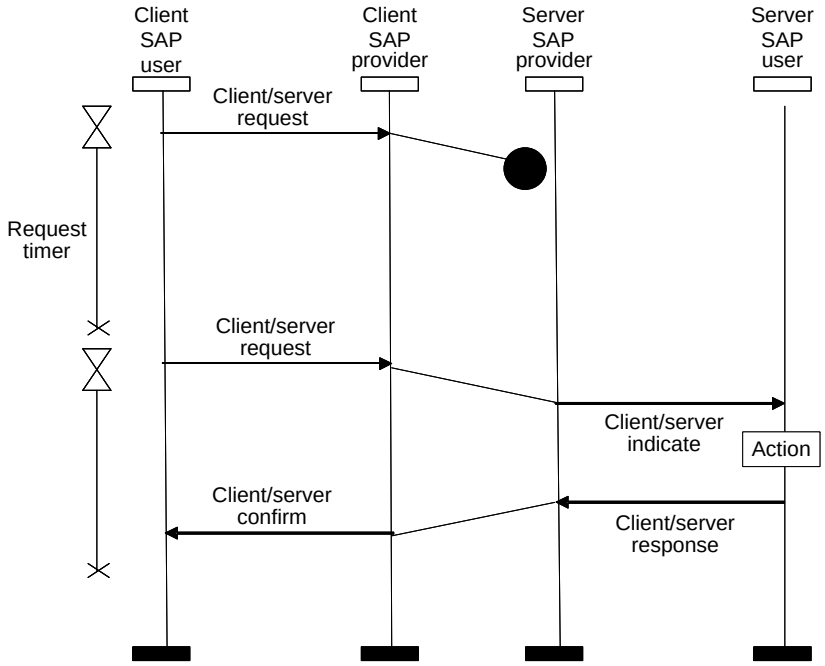


Figure 128 – Client/server model four-part interactions: Request delivery failure

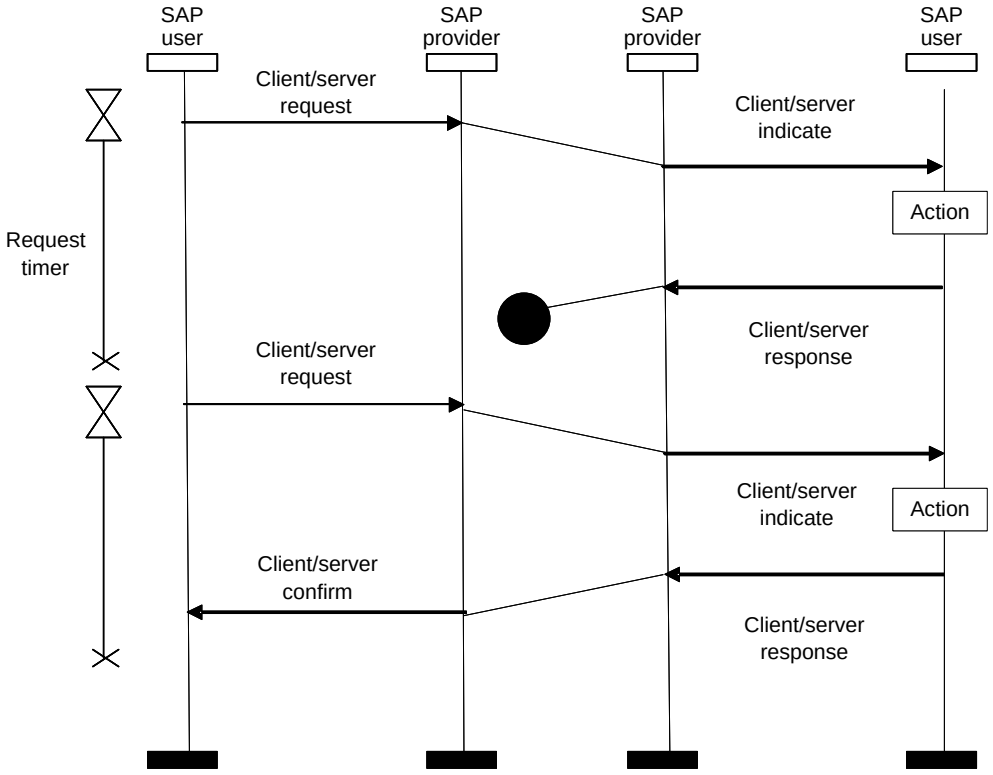


Figure 129 – Client/server model four-part interactions: Response delivery failure

When a client/server association is secured, a security session is involved. To optimize elimination of connections that the UAL process knows are no longer required, a local interface to terminate the contract may be employed. Contract termination may be used to release a security session (if one exists).

To initiate communication, the client requests to send a message to a server. The client specifies the local contract identifier which indicates the server's IPv6Address. The communication also identifies the particular source application and object making the request, the destination application and object intended to receive the request, as well as the actual service instance specific request information.

The server sends a response to the client specifying its local contract identifier, which indicates the client's IPv6Address. The communication also specifies sufficient information to deliver the response to the appropriate application object and to collate the response with the original request.

A communication contract shall be established between client and server to carry the client's request, and correspondingly from the server to the client to carry the server response.

Acting in the role of a client, a client may send requests to a server. Acting in the role of a server, a server may send responses to a client. The client is responsible for server timeout response and transactional integrity.

A simple server might support as few as one outstanding transaction with a particular client. If an extended delay occurs in receiving a response, the client may, for example, timeout and resend the request. If this occurs, duplicate responses may be received. If a server has resources to support multiple outstanding transactions with a client, requests and responses may arrive out of order. To support this situation, a request identifier is used to enable request/response collation.

If there is a need for multiple client or multiple server messages as part of a communication sequence, the implementation may consider employing ASL concatenation. Beyond concatenation, streaming of messages is an application process-specific responsibility outside the scope of this standard.

Communications characteristics for client/server interactions such as response timeout are local matters, beyond the scope of this standard. For example, they may be fixed by device construction, or determined by an application program within the device, or configured for the device on a per-application process basis or even a per contract basis. Client/server interactions are usually used for configuration (such as process control related configuration or management object configuration) and ad-hoc exchange of information.

Client/server communications should not interfere with scheduled communications as it is essential that transmission bandwidth be allocated to support client/server messaging communication contracts. The intent is to ensure the ability to reconfigure a device.

Occasionally, only a single client/server exchange is required. This may entail substantial overhead in route and security establishment. At other times, multiple client/server exchanges occur between the same endpoints.

NOTE Any alteration of communication routes (for example, to compensate for interference) occurs transparently to the AL.

The client specifies the desired message priority for service requests; the server specifies the desired message priority for service responses.

Higher priority messages should ideally move to the front of prioritized communication protocol suite message queues supporting client/server communications. If possible, client/server message bandwidth allocation on the network should grant access first to higher priority message requests. Security is presumed to be on a contract basis, so per-message security is not provided.

Further considerations for transmission back-off, such as based on network congestion are an overall device responsibility, and are not a specific responsibility of the AL.

12.17.4.2 Client/server services

12.17.4.2.1 General

The following services are provided as client/server communications:

- read;
- write;
- execute; and
- tunnel.

NOTE Tunnel as a two-part primitive is also useful for source/sink communication.

12.17.4.2.2 Service feedback codes

Four-part client/server services provide a service feedback code to indicate the result of the service from the viewpoint of the server. A range of codes is reserved for vendor-specific additions.

12.17.4.3 Read

12.17.4.3.1 General

The read service is used to read an attribute of an object from a UAL process.

Table 275 defines the read service primitives.

Table 275 – Read service

Parameter name	Request	Indication	Response	Confirm
Argument	M	M	—	—
Service contract identifier	M	—	—	—
Priority	M	—	—	—
Discard eligible	M	—	—	—
End-to-end transmission time	—	M	—	—
Forward congestion notification	—	M	—	—
Server T-port	M		—	—
Server TDSAP	—	M	—	—
Server object identifier	M	M(=)	—	—
Client IPv6Address	—	M	—	—
Client TDSAP	M	—	—	—
Client T-port	—	M	—	—
Client object identifier	M	M(=)	—	—
Application request ID	M	M(=)	—	—
Data to be read	M	M(=)	—	—
Attribute identifier	M	M(=)	—	—
Attribute index(es)	C	C(=)	—	—

Parameter name	Request	Indication	Response	Confirm
Result	—	—	M	M
Service contract identifier	—	—	M	—
Priority	—	—	M	—
Discard eligible	—	—	M	—
End-to-end transmission time	—	—	—	M
Forward congestion notification	—	—	—	M
Forward congestion notification echo	—	—	M	M(=)
Server IPv6Address	—	—	—	M
Server TDSAP	—	—	M	—
Server T-port	—	—	—	M
Server object identifier	—	—	M	M(=)
Client T-port	—	—	M	—
Client TDSAP	—	—	—	M
Client object identifier	—	—	M	M(=)
Application request ID	—	—	M	M(=)
Value read	—	—	M	M(=)
Service feedback code	—	—	M	M(=)
Value size	—	—	C	C(=)
Data value	—	—	C	C(=)

12.17.4.3.2 Argument

12.17.4.3.2.1 Service contract identifier

See 12.17.3.2.2.1.

12.17.4.3.2.2 Priority

See 12.17.3.2.2.2.

12.17.4.3.2.3 Discard eligible

See 12.17.3.2.2.3.

12.17.4.3.2.4 End-to-end transmission time

See 12.17.3.2.2.4.

12.17.4.3.2.5 Forward congestion notification

This parameter indicates if the request has encountered network congestion on its path from the client to the server.

12.17.4.3.2.6 Server T-port

This parameter identifies the server UAP's associated T-port.

12.17.4.3.2.7 Server TDSAP

This parameter identifies the server TDSAP associated with the server T-port.

12.17.4.3.2.8 Server object identifier

This parameter identifies a server object from which data is desired to be read.

12.17.4.3.2.9 Client/source address

This parameter identifies the IPv6Address for the client of this request.

12.17.4.3.2.10 Client/source TDSAP

This parameter identifies the client or source UAP's associated TDSAP. The UAP contains the object originating the request.

12.17.4.3.2.11 Client T-port

This parameter identifies the client UAP's associated T-port.

12.17.4.3.2.12 Client object identifier

This parameter identifies the client object that is initiating the service request.

12.17.4.3.2.13 Application request identifier

An identifier provided by the UAP to uniquely represent this request.

12.17.4.3.2.14 Data to be read

This parameter identifies the data values that the client desires to read.

12.17.4.3.2.15 Attribute identifier

This parameter identifies the attribute of the server object, the value of which is desired to be read.

12.17.4.3.2.16 Attribute index/indices

This parameter identifies the index/indices for the information of interest from the attribute. There may be:

- no index, such as for a scalar value;
- one index, for example, to access
  - an element of a singly-dimensioned array, or
  - a member of a standard data structure; or
- two indices, for example, to access
  - an element of a doubly-dimensioned array, or
  - a member of a data structure that is contained in a singly-dimensioned array of identical standard data structures, or
  - a singly-dimensioned of a doubly-dimensioned array, or
  - a singly-dimensioned slice of a singly-dimensioned array of identical standard data structures, extracting as a singly-dimensioned array a cross-sectional slice of a single member through those identical data structures.

12.17.4.3.3 Result

12.17.4.3.3.1 Service contract identifier

See 12.17.3.2.2.1.

12.17.4.3.3.2 Priority

See 12.17.3.2.2.2.

12.17.4.3.3.3 Discard eligible

See 12.17.3.2.2.3.

12.17.4.3.3.4 End-to-end transmission time

See 12.17.3.2.2.4.

12.17.4.3.3.5 Forward congestion notification

See 12.17.4.3.2.5.

12.17.4.3.3.6 Forward congestion notification echo

This parameter indicates if the service request encountered network congestion on its path from the client to the server.

12.17.4.3.3.7 Server IPv6Address

This parameter identifies the IPv6Address for the server for this request.

12.17.4.3.3.8 Server TDSAP

See 12.17.4.3.2.7.

12.17.4.3.3.9 Server T-port

See 12.17.4.3.2.6.

12.17.4.3.3.10 Server object identifier

See 12.17.4.3.2.8.

12.17.4.3.3.11 Client T-port

The UAP contains the object originating the request. See 12.17.4.3.2.11.

12.17.4.3.3.12 Client TDSAP

The UAP contains the object originating the request. See 12.17.4.3.2.10.

12.17.4.3.3.13 Client object identifier

See 12.17.4.3.2.12.

12.17.4.3.3.14 Application request identifier

This parameter is an identifier provided by the client to uniquely represent this request.

12.17.4.3.3.15 Value read

The value read indicates the result of the requested operation, and if the read was successful, the size and value of the object attribute to be read.

12.17.4.3.3.16 Service feedback code

The service feedback code indicates if the requested operation was successful or not. If not successful, it provides information indicating why it was not successful.

12.17.4.3.3.17 Value size

Value size indicates the number of octets contained in the data value. It is present if and only if the corresponding service feedback code indicates success.

12.17.4.3.3.18 Data value

Data value is the data value that was read from the identified server object, attribute, and attribute index. It is present if and only if the corresponding service feedback code indicates success, and the value size is non-zero.

12.17.4.4 Write

12.17.4.4.1 General

The write service is used to write a value or set of values to one or more attributes of one or more objects in an application process.

A write to a structure containing both writeable and read-only elements is permitted. In this situation, the read-only elements shall be unaffected.

Table 276 defines the write service primitives.

Table 276 – Write service

Parameter name	Request	Indication	Response	Confirm
Argument	M	M	—	—
Service contract identifier	M	—	—	—
Priority	M	—	—	—
Discard eligible	M	—	—	—
End-to-end transmission time	—	M	—	—
Forward congestion notification	—	M	—	—
Server T-port	M	—	—	—
Server TDSAP	—	M	—	—
Server object identifier	M	M(=)	—	—
Client IPv6Address	—	M	—	—
Client TDSAP	M	—	—	—
Client T-port	—	M	—	—
Client object identifier	M	M(=)	—	—
Application request ID	M	M(=)	—	—
Data to write	M	M(=)	—	—
Attribute identifier	M	M(=)	—	—
Attribute index(es)	M	M(=)	—	—
Value size	M	M(=)	—	—
Data value	M	M(=)	—	—
Result	—	—	M	M
Service contract identifier	—	—	M	—
Priority	—	—	M	—
Discard eligible	—	—	M	—
End-to-end transmission time	—	—	—	M
Forward congestion notification	—	—	—	M
Forward congestion notification echo	—	—	M	M(=)
Server IPv6Address	—	—	—	M
Server TDSAP	—	—	M	—
Server T-port	—	—	—	M(=)
Server object identifier	—	—	M	M(=)
Client T-port	—	—	M	—
Client TDSAP	—	—	—	M
Client object identifier	—	—	M	M(=)
Application request ID	—	—	M	M(=)
Service feedback code	—	—	M	M(=)

12.17.4.4.2 Argument

12.17.4.4.2.1 Service contract identifier

See 12.17.3.2.2.1.

12.17.4.4.2.2 Priority

See 12.17.3.2.2.2.

12.17.4.4.2.3 Discard eligible

See 12.17.3.2.2.3.

12.17.4.4.2.4 End-to-end transmission time

See 12.17.3.2.2.4.

12.17.4.4.2.5 Forward congestion notification

See 12.17.4.3.3.6.

12.17.4.4.2.6 Server T-port

See 12.17.4.3.2.6

12.17.4.4.2.7 Server TDSAP

See 12.17.4.3.2.7

12.17.4.4.2.8 Server object identifier

See 12.17.4.3.2.8

12.17.4.4.2.9 Client/source address

This parameter identifies the client or source UAP's associated IPv6Address. The UAP contains the object originating the request.

12.17.4.4.2.10 Client/source TDSAP

This parameter identifies the client or source UAP's associated TDSAP. The UAP contains the object originating the request.

12.17.4.4.2.11 Client T-port

See 12.17.4.3.2.11.

12.17.4.4.2.12 Client object identifier

See 12.17.4.3.2.12.

12.17.4.4.2.13 Application request identifier

An identifier provided by the UAP to uniquely represent this request.

12.17.4.4.2.14 Data to write

This parameter identifies the target attribute and data value that the client desires to write.

12.17.4.4.2.15 Attribute identifier

This parameter identifies the attribute of the server object, the value of which is desired to be read.

12.17.4.4.2.16 Attribute index/indices

This parameter identifies the index/indices for the information of interest from the attribute. See 12.17.4.3.2.16.

12.17.4.4.2.17 Value size

Value size indicates the number of octets contained in data value.

12.17.4.4.2.18 Data value

Data value is the data value that is desired to be written to the identified server object, attribute, and attribute index.

12.17.4.4.3 Result

12.17.4.4.3.1 Service contract identifier

See 12.17.3.2.2.1.

12.17.4.4.3.2 Priority

See 12.17.3.2.2.2.

12.17.4.4.3.3 Discard eligible

See 12.17.3.2.2.3.

12.17.4.4.3.4 End-to-end transmission time

See 12.17.3.2.2.4.

12.17.4.4.3.5 Forward congestion notification

See 12.17.4.3.2.5.

12.17.4.4.3.6 Forward congestion notification echo

See 12.17.4.3.3.6.

12.17.4.4.3.7 Server IPv6Address

See 12.17.4.3.3.7.

12.17.4.4.3.8 Server TDSAP

See 12.17.4.3.2.7.

12.17.4.4.3.9 Server T-port

See 12.17.4.3.2.6.

12.17.4.4.3.10 Server object identifier

This parameter identifies a server object to which data is desired to be written.

12.17.4.4.3.11 Client/source T-port

This parameter identifies the client UAP's associated T-port.

12.17.4.4.3.12 Client TDSAP

This parameter identifies the client TDSAP associated with the T-port. The UAP contains the object originating the request.

12.17.4.4.3.13 Client object identifier

See 12.17.4.3.2.12.

12.17.4.4.3.14 Application request identifier

See 12.17.4.3.3.14.

12.17.4.4.3.15 Service feedback code

See 12.17.4.3.3.16.

12.17.4.5 Execute

12.17.4.5.1 General

The execute service is used to execute a network visible method on an object.

NOTE Use of the execute service to establish a callback method is one way to provide a server with adequate time for a delayed response, providing information back to the client via a callback, rather than having to provide timely execution results in the response.

Table 277 defines the execute service primitives.

Table 277 – Execute service

Parameter name	Request	Indication	Response	Confirm
Argument	M	M	—	—
Service contract identifier	M	—	—	—
Priority	M	—	—	—
Discard eligible	M	—	—	—
End-to-end transmission time	—	M(=)	—	—
Forward congestion notification	—	M	—	—
Server T-port	M	—	—	—
Server TDSAP	—	M	—	—
Server object identifier	M	M(=)	—	—
Client IPv6Address	—	M	—	—
Client TDSAP	M	—	—	—
Client T-port	—	M	—	—
Client object identifier	M	M(=)	—	—
Application request ID	M	M(=)	—	—
Method to execute	M	M(=)	—	—
Method identifier	M	M(=)	—	—
Size of input parameters	M	M(=)	—	—
Input parameters	C	C(=)	—	—
Result	—	—	M	M
Service contract identifier	—	—	M	—
Priority	—	—	M	—
Discard eligible	—	—	M	—
End-to-end transmission time	—	—	—	M
Forward congestion notification	—	—	—	M
Forward congestion notification echo	—	—	M	M(=)
Server IPv6Address	—	—	—	M
Server TDSAP	—	—	M	—
Server T-port	—	—	—	M
Server object identifier	—	—	M	M(=)
Client T-port	—	—	M	—
Client TDSAP	—	—	—	M
Client object identifier	—	—	M	M(=)
Application request ID	—	—	M	M(=)
Execution result	—	—	M	M(=)
Service feedback code	—	—	M	M(=)
Size of output parameters	—	—	M	M(=)
Output parameters	—	—	C	C(=)

12.17.4.5.2 Argument

12.17.4.5.2.1 Service contract identifier

See 12.17.3.2.2.1.

12.17.4.5.2.2 Priority

See 12.17.3.2.2.2.

12.17.4.5.2.3 Discard eligible

See 12.17.3.2.2.3.

12.17.4.5.2.4 End-to-end transmission time

See 12.17.3.2.2.4.

12.17.4.5.2.5 Forward congestion notification

See 12.17.4.3.2.5.

12.17.4.5.2.6 Server T-port

See 12.17.4.3.2.6.

12.17.4.5.2.7 Server TDSAP

See 12.17.4.3.2.7.

12.17.4.5.2.8 Server object identifier

See 12.17.4.3.2.8.

12.17.4.5.2.9 Client/source address

See 12.17.4.3.2.9

12.17.4.5.2.10 Client/source TDSAP

See 12.17.4.3.2.10.

12.17.4.5.2.11 Client T-port

See 12.17.4.3.2.11.

12.17.4.5.2.12 Client object identifier

See 12.17.4.3.2.12.

12.17.4.5.2.13 Application request identifier

See 12.17.4.3.2.13.

12.17.4.5.2.14 Method identifier

This parameter identifies the method of the server object that is desired to be executed.

12.17.4.5.2.15 Size of input parameters

Size of input parameters indicates the number of octets contained in input parameters.

NOTE Execute requests and responses include the size in octets of the contained parameter stream to enable parsing (this is especially useful in APDU concatenation scenarios).

12.17.4.5.2.16 Input parameters

The input parameters' string is an octet string that contains the input parameters for the method that is being requested to be executed. This is present if and only if size of input parameters is present and has a value greater than zero.

12.17.4.5.3 Result

12.17.4.5.3.1 Service contract identifier

See 12.17.3.2.2.1.

12.17.4.5.3.2 Priority

See 12.17.3.2.2.2.

12.17.4.5.3.3 Discard eligible

See 12.17.3.2.2.3.

12.17.4.5.3.4 End-to-end transmission time

See 12.17.3.2.2.4.

12.17.4.5.3.5 Forward congestion notification

See 12.17.4.3.2.5.

12.17.4.5.3.6 Forward congestion notification echo

See 12.17.4.3.3.6.

12.17.4.5.3.7 Server IPv6Address

See 12.17.4.3.3.7.

12.17.4.5.3.8 Server TDSAP

See 12.17.4.3.2.7.

12.17.4.5.3.9 Server T-port

See 12.17.4.3.2.6.

12.17.4.5.3.10 Server object identifier

See 12.17.4.4.3.10.

12.17.4.5.3.11 Client/source T-port

See 12.17.4.4.3.11.

12.17.4.5.3.12 Client TDSAP

See 12.17.4.4.3.12.

12.17.4.5.3.13 Client object identifier

See 12.17.4.3.2.12.

12.17.4.5.3.14 Application request identifier

See 12.17.4.3.3.14.

12.17.4.5.3.15 Execution result

This contains the result of the method execution service request.

12.17.4.5.3.16 Service feedback code

The service feedback code indicates if the corresponding method execution was successful or not. If not successful, it provides information indicating why it was not successful.

12.17.4.5.3.17 Size of output parameters

Size of output parameters indicates the number of octets contained in output parameters.

12.17.4.5.3.18 Output parameters

The output parameters' string is an octet string that contains the output parameters for the method that was executed. This is present if and only if size of output parameters is present and has a value greater than zero.

12.17.5 Unscheduled acyclic queued unidirectional messages (source/sink)

12.17.5.1 General

Unscheduled acyclic queued unidirectional messaging is also sometimes referred to as source/sink messaging. This interaction type is used for alerts. Messages sent using this protocol are queued by the lower communication layers for transmission. Message receipt is unconfirmed. There is no application process flow or rate control or lost message detection for this mode of interaction. Like client/server communications, these communications require use of a communication contract, and specify message priority on a per-message basis.

Bandwidth for source/sink communications is not considered dedicated, but rather is considered to come from non-dedicated (i.e., shared) bandwidth.

Unscheduled acyclic unidirectional interactions in this standard support on-demand one-to-one queued message distribution. Alert reports for network communication are always issued from one initiator, the ARMO, and are always sent to one type of message recipient, an alert-receiving object (ARO).

Acknowledgment of reception of an individual alert may only be issued from one alert report recipient (ARO), and is sent to the (ARMO) object that reported the alert.

The following services are provided to support unscheduled acyclic queued unidirectional message communications:

- AlertReport;
- AlertAcknowledge; and
- Tunnel.

The Tunnel service is included as a source/sink service so that it will be able to take advantage of multi-cast capabilities in the future. Such a potential development is a subject for future standardization.

12.17.5.2 AlertReport service

12.17.5.2.1 General

AlertReport is used to report an alert using queued unidirectional communication services. The content of the alert report depends on the type of alert being reported and the category of the alert. AlertReports may be retried until an AlertAcknowledge for the AlertReport has been received.

Figure 130, Figure 131, and Figure 132 indicate alert reporting message sequencing.

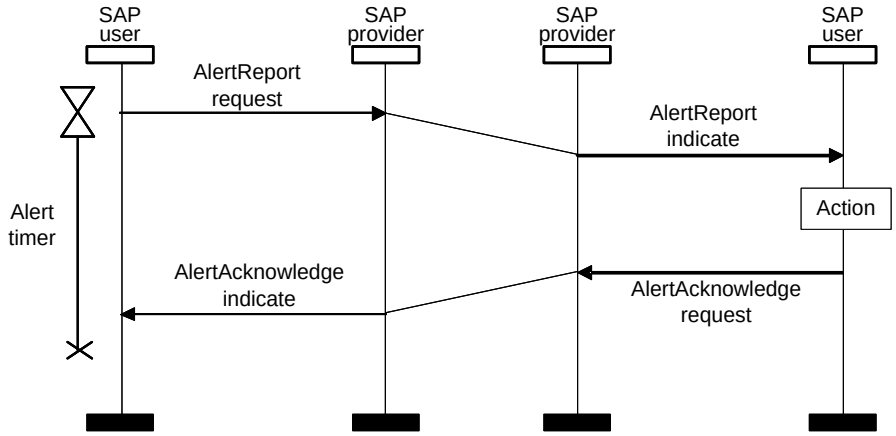


Figure 130 – AlertReport and AlertAcknowledge, delivery success

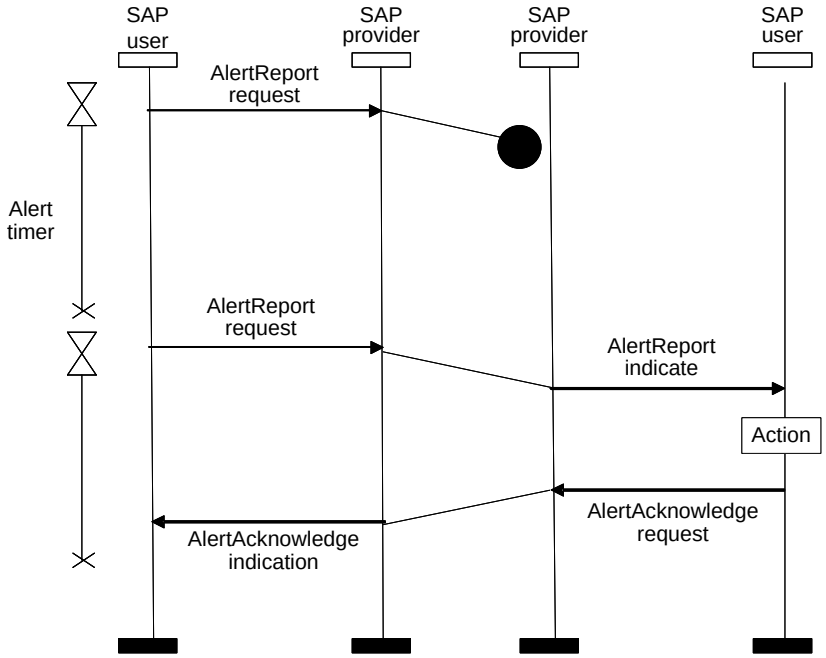


Figure 131 – AlertReport, delivery failure

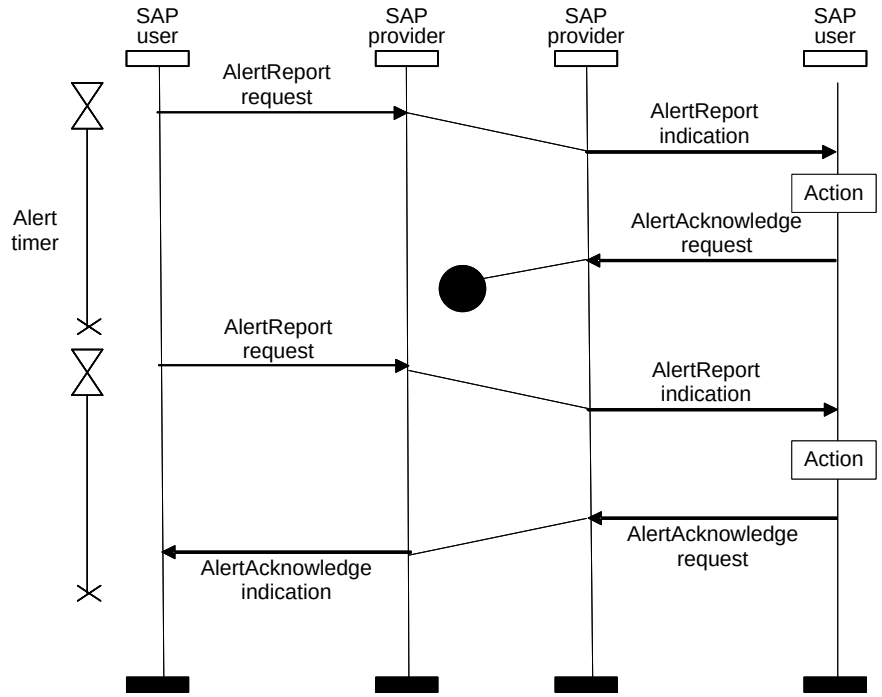


Figure 132 – AlertReport, acknowledgment failure

NOTE 1 AlertReport timeout/retry policy is defined on the ARMO. See 6.2.7.2 for further details.

Alert reporting employs two separate two-part application communication services. To report an alert, the AP uses the AlertReport service. In this version of the standard, the recipient of an AlertReport shall acknowledge the AlertReport by using the AlertAcknowledge service.

NOTE 2 The use of two separate services, AlertReport and AlertAcknowledgment, enable a single alert to be sent to multiple destinations in a future revision to this standard.

Monitoring and checking for acknowledgment, as well as re-reporting an alert condition for which acknowledgment has not been received is the responsibility of the alert reporting device. Re-reporting an alert that is no longer prevalent, and for which an AlertAcknowledge indication has not been received, is a local matter. For example, if a diagnostic situation occurs and an alert is reported, and then the reporting device reboots such that the diagnostic situation is no longer prevailing, the device might not re-report the diagnostic alert that was in effect prior to reboot even though no AlertAcknowledge was received.

AlertReport employs the same communication model as a two-part client/server primitive. Table 278 defines the service primitives for the AlertReport service.

Table 278 – AlertReport service

Parameter name	Request	Indication
Argument	M	M
Service contract identifier	M	—
Priority	M	—
Discard eligible	M	—
End-to-end transmission time	—	M
ARMO TDSAP	M	—
ARMO T-port	—	M
ARMO	M	M(=)
Sink T-port	M	—
Sink TDSAP	—	M
Sink object identifier	M	M(=)
Individual alert report	M	M(=)
Individual alert identifier	M	M(=)
Alert detector T-port	M	M(=)
Alert detector object	M	M(=)
Detection time	M	M(=)
Alert class	M	M(=)
Alarm direction	C	C(=)
Alert category	M	M(=)
Alert priority	M	M(=)
Alert type	M	M(=)
Associated-data size	M	M(=)
Associated data	O	O(=)

12.17.5.2.2 Argument

12.17.5.2.2.1 Service contract identifier

See 12.17.3.2.2.1.

12.17.5.2.2.2 Priority

See 12.17.3.2.2.2.

12.17.5.2.2.3 Discard eligible

See 12.17.3.2.2.3.

12.17.5.2.2.4 End-to-end transmission time

See 12.17.3.2.2.4.

12.17.5.2.2.5 Alert reporting management object TDSAP

This parameter indicates the TDSAP of the application which is issuing this alert report.

12.17.5.2.2.6 Alert reporting management object T-port

This parameter indicates the T-port of the application which is issuing this alert report.

12.17.5.2.2.7 Alert reporting management object

This parameter represents the object identifier of the ARMO that is reporting the alert.

12.17.5.2.2.8 Sink T-port

This parameter identifies the sink UAP's associated T-port.

12.17.5.2.2.9 Sink TL data service access point

This parameter indicates the TDSAP corresponding to sink T-port.

12.17.5.2.2.10 Sink object identifier

This parameter specifies the destination sink object in the application to which this service request is to be sent.

12.17.5.2.2.11 Alert source IPv6Address

This parameter identifies the IPv6Address of the source of this request.

12.17.5.2.2.12 Alert source TDSAP

This parameter identifies the source UAP's associated TDSAP.

12.17.5.2.2.13 Source T-port

This parameter identifies the transmitting application source UAP associated with the T-port.

12.17.5.2.2.14 Individual alert report

This parameter contains an individual alert being reported by this service invocation.

12.17.5.2.2.15 Individual alert identifier

This parameter uniquely identifies the individual alert report. Separate identifier value sequences for the alert reporting categories shall be maintained. The value of this parameter shall be monotonically increasing, and shall wrap around when the maximum value is reached. It is included when an individual alert report is acknowledged. It also is used by an alert receiver to determine if an alert report or a set of alert reports of a particular category have been missed. If a missed report condition is detected, an alarm recovery operation should be performed. Refer to the Alarm_Regen attributes of the ARMO in 6.2.7.2 for further details on triggering the regeneration.

12.17.5.2.2.16 Alert source transport port

This parameter identifies the UAP containing the object that detected the alert via its associated T-port.

12.17.5.2.2.17 Alert source object

The alert source object indicates the object instance that detected the alarm condition.

NOTE The alert reporting management object reports alert conditions detected by one or more alert detecting objects.

12.17.5.2.2.18 Detection time

This parameter specifies the time at which the alert condition was detected. This value indicates the network time at which the alert was detected. How time information is made available to an application reporting an alert is a device local matter, not specified by this standard.

NOTE Translating network time to social time (wall clock time), when desired, is performed in the gateway. See 5.6 for further details.

12.17.5.2.2.19 Alert class

This parameter indicates if this is an event (stateless) or alarm (state-oriented) type of alert.

12.17.5.2.2.20 Alarm direction

For alerts that are state-oriented (alarms), this indicates if the report is for an alarm condition, or a return to normal from an alarm condition.

12.17.5.2.2.21 Alert category

Alert category indicates if the alert is a device diagnostic alert, a communication diagnostic alert, a security alert, or a process alert.

12.17.5.2.2.22 Alert priority

Alert priority is a value that suggests the importance of the alert. A larger value implies a more important alert. Host systems map device priorities into host alert priorities that usually include urgent, high, medium, low, and journal. The recommended mapping of alert priority values into these categories is as follows:

- 0..2: journal
- 3..5: low
- 6..8: medium
- 9..11: high
- 12..15: urgent

Since the interpretation of alert priorities occurs primarily in the originating and intended receiving devices, other assignments that reflect a differing categorization are permitted.

12.17.5.2.2.23 Alert type

Alert type provides additional information regarding the alert, specific to the alert category.

12.17.5.2.2.24 Associated-data size

Associated-data size specifies the size of any alert-specific data conveyed with the alert.

12.17.5.2.2.25 Associated data

Associated data provides a means of conveying alert-specific data.

12.17.5.3 AlertAcknowledge service

12.17.5.3.1 General

AlertAcknowledge is a two-part service that is used to acknowledge an individual alert to an alert reporting management object. For unicast alert reports, receipt of an AlertAcknowledge shall result in the ceasing of AlertReport retry requests for the corresponding individual alert.

An AlertAcknowledge shall be sent for every AlertReport received.

NOTE If a duplicate AlertReport has been received, either the application that sent the AlertReport did not receive the AlertAcknowledge within its timeout/retry time, or the AlertAcknowledgment message was not received. Since the application sending the AlertAcknowledge does not know which situation occurred, a duplicate acknowledgment is sent.

The AlertAcknowledge service is described in Table 279.

Table 279 – AlertAcknowledge service

Parameter name	Request	Indication
Argument	M	M
Service contract identifier	M	—
Priority	M	—
Discard eligible	M	—
End-to-end transmission time	—	M
Source IPv6Address	—	M
Source TDSAP	M	—
Source T-port	—	M
Source object identifier	M	M(=)
Destination transport port	M	—
Destination TDSAP	—	M
Destination object identifier	M	M(=)
Individual alert identifier	M	M(=)

12.17.5.3.2 Argument

12.17.5.3.2.1 Service contract identifier

See 12.17.3.2.2.1.

12.17.5.3.2.2 Priority

See 12.17.3.2.2.2.

12.17.5.3.2.3 Discard eligible

See 12.17.3.2.2.3.

12.17.5.3.2.4 End-to-end transmission time

See 12.17.3.2.2.4.

12.17.5.3.2.5 Source IPv6Address

This parameter identifies the IPv6Address for the source of this request.

12.17.5.3.2.6 Source TDSAP

This parameter identifies the service primitive source AP's associated TDSAP. The TDSAP maps 1-to-1 to a UAP.

12.17.5.3.2.7 Source T-port

This parameter identifies the transmitting application source UAP associated with the T-port.

12.17.5.3.2.8 Source object identifier

This parameter identifies the object that is initiating the alert acknowledgment.

12.17.5.3.2.9 Destination T-port

This parameter identifies the application process to receive the alert acknowledgment as a single application is associated with a T-port.

12.17.5.3.2.10 Destination TDSAP

This parameter identifies the TDLE SAP corresponding to the destination transport port.

12.17.5.3.2.11 Destination object identifier

This parameter identifies the object in the application process of the device to receive the acknowledgment.

12.17.5.3.2.12 Individual alert identifier

This parameter identifies the individual alert that is being acknowledged.

12.17.6 Client/server and source/sink commonalities

12.17.6.1 Individual or concatenated messaging for client/server and/or source/sink

Client/server and source/sink messages may be sent as an individual transport service data unit (TSDU), or may be concatenated together within a single TSDU. Concatenation supports both four-part primitive messages (requests and responses) and two-part primitive messages (requests only). Concatenation allows client/server messages to be combined in the TSDU with source/sink messages. How APDU concatenation is determined and accomplished is a device local matter. It is recommended that concatenations refrain from including more than two services, as this may result in more bursty communication.

NOTE 1 The discussion of stretch acknowledgment violation in IETF RFC 2525 provides background on message acknowledgment concatenation and the ramifications of having more than two such acknowledgments in a PDU.

NOTE 2 Publish/subscribe already supports including multiple values from a UAP in a single message. See 12.15.2.5 and 12.15.2.6 for further details.

Concatenation may be used to reduce transmission overhead and/or deliver a set of messages to the corresponding ASL as a unit.

All messages within the concatenation shall have a common message priority, and shall indicate communication via a common communication contract.

The number of ASL services that may be concatenated by an ASL building the concatenated PDU is limited by the maximum APDU size corresponding to the communication contract to be used for the messages.

The ASL receiving a concatenated APDU is required to parse and handle each APDU individually from start to finish until the end of the TSDU has been met. If a protocol error is detected during parsing of a concatenated APDU, a single malformed APDU error is indicated and the remaining portion of the APDU shall be discarded.

An ASL concatenation of services may contain:

- homogeneous ASL service primitives (e.g., all service requests); or
- heterogeneous ASL service primitives (e.g., for client/server flows, requests and responses, which may be mixed);
- homogeneous application communication flow primitives (e.g., all client/server); or
- heterogeneous application communication flow type primitives (e.g., client/server and source/sink).

The AL itself makes no requirements on how responses to services included in a concatenation are returned; determination of such responses is at the discretion of the receiving application. For example, if a concatenated client/server request contains service requests (A, B, C) and another concatenated client/server service request contains service requests (D, E), the client/server responses for these may:

– not be required (e.g., A, B, and D may be four-part services that require a response, but C and E may be two-part services that do not require a response);
– not be concatenated at all, and returned in any order (e.g., response A, response B, response E, response D, response C, all in separate APDUs);
– be partially concatenated, in any order (e.g., response returned may be B, C in one APDU, A in another APDU);
– employ a single APDU to respond to a concatenated request, but responses may be concatenated in any order (e.g., response returned may be concatenated as BCA);
– be fully concatenated in the same order (e.g., response returned concatenated as ABC);
– be entirely differently concatenated than the requests received (e.g., response may be returned BD, ACE).

How and when an ASL initiating a communication determines when to create a concatenation as well as when to deliver the concatenation to the lower communication protocol suite for conveyance is a local matter. However, this standard shall specify the overall structure of a TSDU containing concatenated APDUs.

This standard does not prescribe the order in which services included in concatenated messaging are handled by the destination. Thus the order of responses is not required to be the same as the order of the requests included in the concatenation.

NOTE 3 An implementation that defines local services to bracket concatenated constructions is able to provide further control over concatenation content.

Figure 133 illustrates a concatenated response for multiple outstanding write requests with no message loss. Timeouts are described in 12.12.4.2.2.1.

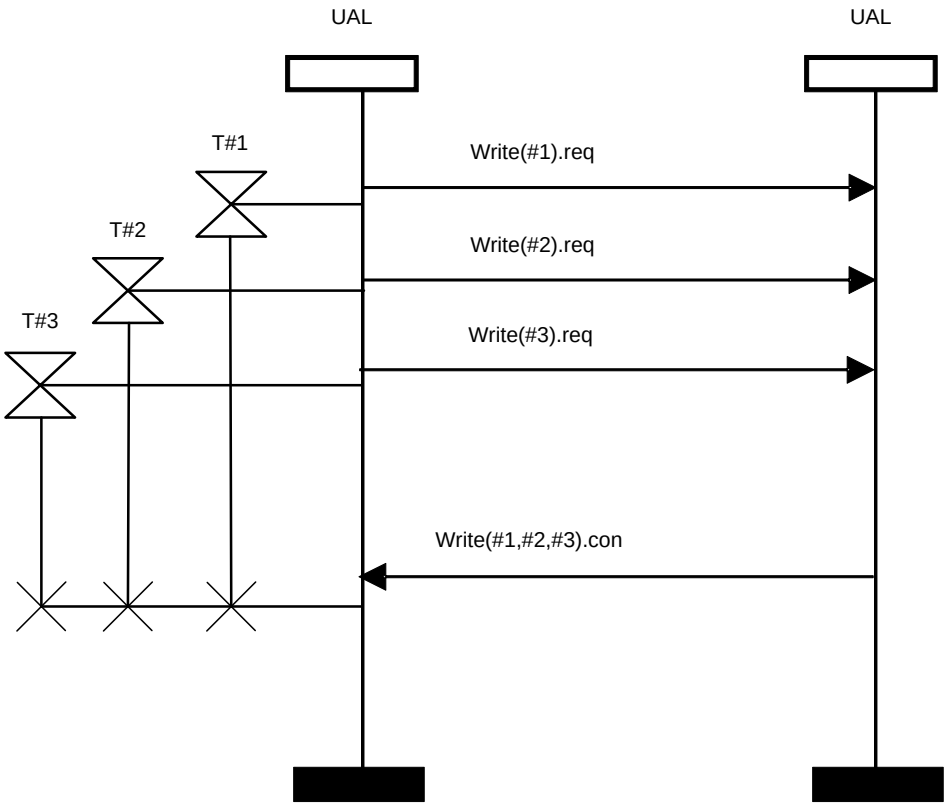


Figure 133 – Concatenated response for multiple outstanding write requests (no message loss)

12.17.6.2 Application sublayer common services for client/server and source/sink messaging – Tunnel

12.17.6.2.1 General

The tunnel service may use either a two-part (source/sink) or a four-part (client/server) communication service primitive model.

Information to enable matching service primitives, effecting appropriate application level handling of error situations such as duplicate message detection and detection of out-of-order delivery, etc., shall be the responsibility of the application process to include within the non-native payload of the tunnel messages.

The tunnel service can be used to encapsulate both client/server and source/sink communications as defined by this standard. This service identifies a message as destined for a non-native protocol tunnel. The service identifier is required so that the ASL can parse a message and determine whether to pass it on to a legacy protocol tunnel or handle it as a native message. The tunnel service provides a single level of message encapsulation for a protocol tunnel. The non-native APDU is passed through to the destination object specified in the tunnel service request.

A tunneling application may establish the retry policy for two-part (source/sink) tunnel requests that it sends.

Table 280 defines the tunnel service primitives.

Table 280 – Tunnel service

Parameter name	Request	Indication	Response	Confirm
Argument	M	M	—	—
Service contract identifier	M	—	—	—
Priority	M	—	—	—
Discard eligible	M	—	—	—
End-to-end transmission time	—	M	—	—
Forward congestion notification	—	M	—	—
Application destination T-port	M	—	—	—
Application destination TDLE SAP	—	M	—	—
Application destination object identifier	M	M(=)	—	—
Application source IPv6Address	—	M	—	—
Application source TLDE SAP	M	—	—	—
Application source T-port	—	M	—	—
Application source object identifier	M	M(=)	—	—
Payload size (in octets)	M	M(=)	—	—
Tunnel payload data	M	M(=)	—	—
Result	—	—	U	U
Service contract identifier	—	—	M	—
Priority	—	—	M	—
Discard eligible	—	—	M	—
End-to-end transmission time	—	—	—	M
Forward congestion notification	—	—	—	M
Forward congestion notification echo	—	—	M	M(=)
Application destination T-port	—	—	M	—
Application destination TDLE SAP	—	—	—	M
Application destination object identifier	—	—	M	M(=)
Application source IPv6Address	—	—	—	M
Application source TLDE SAP	—	—	M	—
Application source T-port	—	—	—	M(=)
Application source object identifier	—	—	M	M(=)
Payload size (in octets)	—	—	M	M(=)
Tunnel payload data	—	—	M	M(=)

12.17.6.2.2 Argument

12.17.6.2.2.1 Service contract identifier

See 12.17.3.2.2.1.

12.17.6.2.2.2 Priority

See 12.17.3.2.2.2.

12.17.6.2.2.3 Discard eligible

See 12.17.3.2.2.3.

12.17.6.2.2.4 End-to-end transmission time

See 12.17.3.2.2.4.

12.17.6.2.2.5 Forward congestion notification

This parameter indicates if the request has encountered network congestion on its path from the client to the server.

12.17.6.2.2.6 Application destination T-port

This parameter identifies the UAP at the destination for this service request.

12.17.6.2.2.7 Application destination TDSAP

This parameter represents the TDSAP corresponding to the AL transport destination port.

12.17.6.2.2.8 Application destination object identifier

This parameter identifies the object in the receiving application.

12.17.6.2.2.9 Application source IPv6Address

This parameter identifies the IPv6Address for the client of this request.

12.17.6.2.2.10 Application source TDSAP

This parameter identifies the application source UAP's associated TDSAP. The TDSAP maps 1-to-1 to a UAP. The UAP contains the object originating the request in the client.

12.17.6.2.2.11 Application source T-port

This parameter identifies the UAP that is the source for this service request.

12.17.6.2.2.12 Application source object identifier

This parameter indicates the application source object that is originating the tunnel service request.

12.17.6.2.2.13 Payload size

This parameter indicates the number of octets of the tunnel payload parameter.

12.17.6.2.2.14 Tunnel payload data

This parameter represents the data (e.g., legacy protocol APDU) that is to be conveyed to the server object.

12.17.6.2.3 Result

12.17.6.2.3.1 Service contract identifier

See 12.17.3.2.2.1.

12.17.6.2.3.2 Priority

See 12.17.3.2.2.2.

12.17.6.2.3.3 Discard eligible

See 12.17.3.2.2.3.

12.17.6.2.3.4 End-to-end transmission time

See 12.17.3.2.2.4.

12.17.6.2.3.5 Forward congestion notification

This parameter indicates if the service response encountered network congestion on its path from the server to the client.

12.17.6.2.3.6 Forward congestion notification echo

This parameter indicates if the service request encountered network congestion on its path from the client to the server.

12.17.6.2.3.7 Application destination T-port

This parameter identifies the UAP at the destination for this service request.

12.17.6.2.3.8 Application destination TDSAP

This parameter represents the TDSAP corresponding to the AL transport destination port.

12.17.6.2.3.9 Application destination object identifier

This parameter indicates the application destination object that is originating the tunnel service request.

12.17.6.2.3.10 Application source IPv6Address

This parameter identifies the IPv6Address for the client of this request.

12.17.6.2.3.11 Application source TDSAP

This parameter identifies the application source UAP’s associated TDSAP. The TDSAP maps 1-to-1 to a UAP. The UAP contains the object originating the request in the client.

12.17.6.2.3.12 Application source T-port

This parameter indicates the application source object that is originating the tunnel service request.

12.17.6.2.3.13 Application source object identifier

This parameter indicates the application source tunnel object that is originating the tunnel service request.

12.17.6.2.3.14 Payload size

This parameter indicates the number of octets of the tunnel payload parameter.

12.17.6.2.3.15 Tunnel payload data

This parameter represents the data (e.g., legacy protocol APDU) that is to be conveyed to the server object.

12.18 AL flow use of lower layer services

12.18.1 General

All types of messaging (e.g., publication, client/server, source/sink, bulk transfer) and all qualities of service may flow through the common TDSAP for the application process. Table 281 indicates the mapping of the AL flows to the TL services provided.

Table 281 – Application flow characteristics

Application flow type	Buffered or queued	Periodic (scheduled)	Order sensitive	Reliable	Unacknowledged	Message importance		
						High	Medium	Low
Periodic publish/ subscribe	Buffered	Yes	Yes	No	Yes	[a]	[a]	[a]
Client/ server	Queued	No	No	No	Yes	[a]	[a]	[a]
Source/sink	Queued	No	No	No	Yes	[a]	[b]	[a]

[a] The message importance alternative that is selected is the one that applies.

[b] No use case identified.

12.18.2 AL use of TDSAPs

The ASL communicates with the lower layers of the communication protocol suite via TDSAPs. The information communicated to the TL for TDSAP mapping is a subset of the information communicated to the ASL from the UAP at the ASAP. There is one well-known TDSAP in this standard, TDSAP number 0, that is used for communications with the objects represented by the DMAP application.

TDSAPs that are not well known may be associated with one and only one particular application process. That is, there is a one-to-one mapping relationship between a TDSAP and a UAL process. Because TDSAPs are local, remote entities indicate a T-port that represents a corresponding application. T-ports map 1-to-1 to TDSAPs. UAL process/TDSAP/transport data port relationships shall survive application restart.

NOTE Fifteen TDSAPs are available for compressed transmission. See 11.6 for further details.

12.18.3 Mapping AL service primitives to TL service primitives

Table 282 indicates the mapping of application service primitives to transport services.

Table 282 – AL service primitive to TL service primitive mapping

Application service primitive	Transport service	Data conveyed between application sublayer and transport service
Publish.request Read.request, Read.response Write.request, Write.response Execute.request, Execute.response Tunnel.request, Tunnel.response AlertReport.request AlertAcknowledge.request	T-DATA.request	Contract_ID APDU_size APDU Message priority Discard eligibility Source TDSAP Destination T-port NOTE 1 The contract ID indicates destination address, and contract priority. NOTE 2 The source T-port can be determined from the source TDSAP, but is explicitly passed to match the interface provided by the TL.
Publish.indicate Read.indicate, Read.confirmation Write.indicate, Write.confirmation Execute.indicate, Execute.confirmation Tunnel.indicate, Tunnel.confirmation AlertReport.indicate AlertAcknowledge.indicate	T-DATA.indicate	Source IPv6Address Source T-port APDU_size (equivalent to TSDU size) APDU (equivalent to TSDU) Explicit congestion notification (ECN) Destination TDSAP Destination T-port Transport time (one-way end-to-end delivery time, in seconds)

12.19 AL management

12.19.1 General

AL management supports the local DMAP application sublayer management object. Access to the attributes and methods of this object are defined by the ASL. For this standard, the ASL provides access to read a configured value from the ASL-MIB, to write a configured value to the ASL-MIB, and to support reset of the ASL.

12.19.2 Application sublayer handling of malformed application protocol data units

The ASL supports informing the local DMAP of a potential device/communication problem if an AL management-configured threshold is reached within a configured time period, for malformed APDUs received from a particular source device. Some examples of malformed APDUs are:

- APDU size incorrect (too long or too short);
- invalid service identifier; or
- service misuse (e.g., response primitive was indicated in the PDU for a two-part client/server or source/sink service).

The intent of this information is to enable the DMAP to provide information to higher level management. This may be important, for example, to enable detection of a malformed APDU attack.

The ASL may be configured to advise the DMAP whenever a malformed APDU is received.

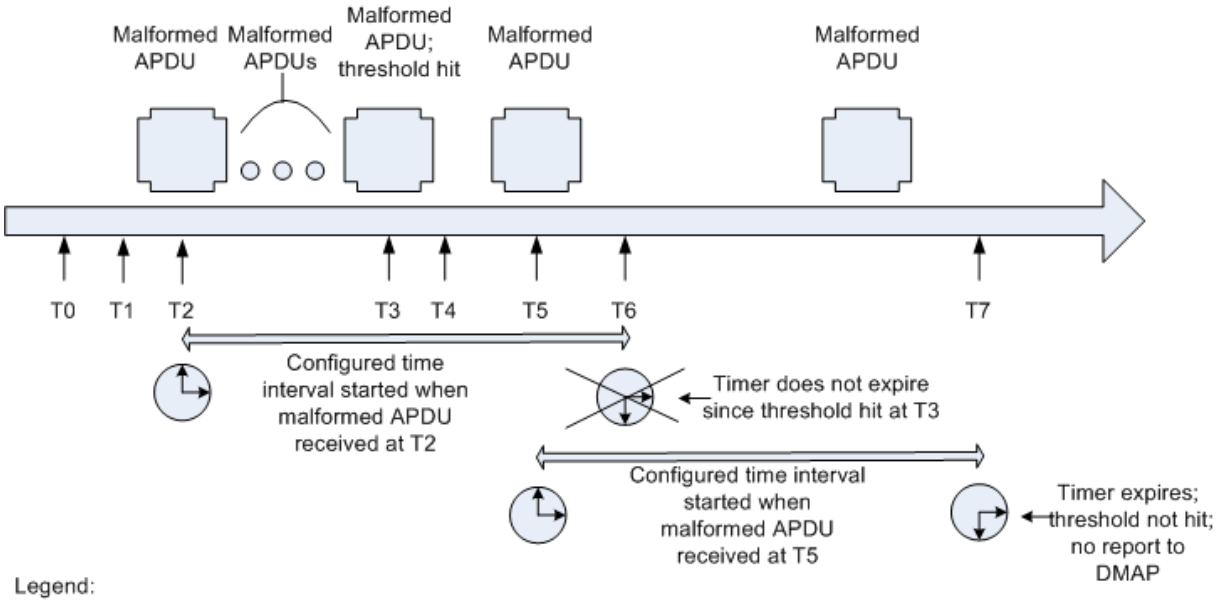
The ASL may be configured with non-zero values for a threshold and a time interval. When so configured, the ASL will maintain individual counters and timers internally for each network source address from which a malformed APDU has been received. Counting commences with the receipt of the first malformed APDU from a device and continues until either the malformed APDU threshold value is reached or the ASL_TimePeriodForMalformedAPDUs expires.

If the malformed APDU threshold value is reached prior to or at the expiration of the configured time interval, the DMAP is advised and the count and time interval for the device are reset.

If the malformed APDU threshold is not reached within the configured time interval, the counters and timers are internally reset.

NOTE How the DMAP is advised of a malformed APDU, or that a threshold has been reached within a specific time interval, is a local matter, and hence is not specified by this standard.

Figure 134 illustrates the handling of malformed APDUs.



Legend:
T0 – malformed APDU threshold is set to non-zero value
T1 – malformed APDU counting interval is set to non-zero value
T2 – first malformed APDU arrives from source device "X", time interval for counting malformed APDUs starts
T3 – threshold of malformed APDUs is reached
T4 – report to DMAP after which internal ASL internal counter resets
T5 -- another malformed APDU arrives again from source device "X"
T6 - original calculation of expiration of time interval for malformed APDUs from device "X" , starting from receipt of malformed APDU at T2
T7 – time interval started at T5 expires without threshold being hit; ASL internal counter resets

NOTE: The order of performing steps T0 and T1 is not important.

Figure 134 – Management and handling of malformed APDUs received from device X

12.19.3 Application sublayer management object attributes

Table 283 describes the attributes supported by the application sublayer management object (ASLMO).

Table 283 – ASLMO attributes *(1 of 2)*

Standard object type name: Application sublayer management object (ASLMO)				
Standard object type identifier: 121				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
ObjectIdentifier	Object key identifier	Unique identifier for the object	Type: Unsigned16 Classification: Constant Default value: 7 Valid range: 7	N/A
Reserved for future use	0	—	—	—
MalformedAPDUsAdvise	1	Indicates if ASL should indicate to local DMAP whenever a malformed APDU is encountered	Type: Boolean Classification: Static Accessibility: Read/write Initial default value : FALSE	If this parameter has a value of TRUE, the ASL shall pass each malformed APDU it receives on to the local DMAP
TimeIntervalForCounting MalformedAPDUs	2	This attribute specifies the time interval for the ASL to count malformed APDUs received from a particular device. Counting occurs from detection of the first malformed APDU from a device. This interval is applied commonly to APDUs from all source IPv6Addresses. The unit of this attribute is seconds	Type: TAITimeDifference Classification: Static Accessibility: Read/write Initial default value : 0 Valid range: 0 ≤ value ≤ to 86 400 s Number of days is not included (it is always equal to zero)	If the time interval expires without the threshold being met, the corresponding internal malformed ASL counter and timer information shall be reset to zero (0)
MalformedAPDUsThreshold	3	Common threshold value to apply to malformed APDUs received from each device	Type: Unsigned16 Classification: Static Accessibility: Read/write Initial default value : 0	If this threshold is met in the specified time interval, a communication alert shall be reported indicating the device that has been sending malformed APDUs. If a threshold value is set while counting is in progress, and the value set is lower than the prior threshold such that the new threshold has been exceeded, a malformed APDU alert shall be reported

Table 284 *(2 of 2)*

Standard object type name: Application sublayer management object (ASLMO)				
Standard object type identifier: 121				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
MalformedAPDUAlertDescriptor	4	Describes how the malformed alert is reported on the network	Type : Alert report descriptor Classification: Static Accessibility: Read/write Default value: [TRUE, 7]	
MaxDevicesForWhichMalformed APDUsCanBeCounted	5	Describes the malformed APDU counting capability of the ASL in terms of the maximum number of devices for which counts can be simultaneously maintained	Type: Unsigned16 Classification: Constant Accessibility: Read only	A minimum value required may be established for example based on the role of the device
Reserved for future use by this standard	6..63	—	—	—

Attributes classified as either constant or static shall be preserved across restarts and power-fails.

12.19.4 Application sublayer management object methods

12.19.4.1 Standard object methods

An ASLMO has the methods defined in Table 284.

Table 284 – Application sublayer management object methods

Standard object type name: Application sublayer management object (ASLMO)		
Standard object type identifier: 121		
Method name	**Method ID**	**Method description**
Null	0	Reserved by this standard for future use
Reset	1	Reset application sublayer
Reserved for future use by this standard	2..127	These method identifiers are reserved for future use by this standard
Implementation-specific use	128..255	These method identifiers are available for implementation-specific use

12.19.4.2 Reset method

Table 285 specifies the reset method primitives.

Table 285 – Reset method

<table>
<tr><th colspan="5">Standard object type name: Application sublayer management object (ASLMO)</th></tr>
<tr><th colspan="5">Standard object type identifier: 121</th></tr>
<tr><th>Method name</th><th>Method ID</th><th colspan="3">Method description</th></tr>
<tr><td rowspan="8">Reset</td><td>1</td><td colspan="3">Reset application sublayer</td></tr>
<tr><td colspan="4">Input arguments</td></tr>
<tr><td>Argument number</td><td>Argument name</td><td>Argument type (data type and size)</td><td>Argument description</td></tr>
<tr><td>1</td><td>ResetType</td><td>Type: Unsigned8

Named values:
0: not used;
1: reset to factory default settings;
2: reset to provisioned settings;
3: warm reset (reset to provisioned settings and any communication contract related information);
4: reset all dynamic data (e.g., related to statistics);
5..255: reserved</td><td>Type of reset desired</td></tr>
<tr><td colspan="4">Output arguments</td></tr>
<tr><td>Argument number</td><td>Argument name</td><td>Argument type (data type and size)</td><td>Argument description</td></tr>
<tr><td colspan="4">None</td></tr>
</table>

12.19.4.3 Input arguments

The ResetType parameter indicates the type of reset desired. The sublayer may be reset to:

- factory default settings;
- only maintain provisioned settings (if any);
- only maintain the set of both provisioned settings and communication contract settings (if any); or
- only all dynamic statistics.

NOTE An example of default factory settings is:

– MalformedAPDUsAdvise configuration parameter, indicating disabled;

– TimeIntervalForCountingMalformedAPDUs configuration parameter, indicating 100 APDUs; and

– MalformedAPDUsThreshold configuration parameter, indicating a zero time interval.

12.19.4.4 Output arguments

There are no output arguments for this method.

12.19.4.5 Response codes

The following feedback codes are valid for this method:

- success;
- invalidArgument; and
- those that are vendor-defined.

12.19.5 Application sublayer management object alerts

Table 286 defines the alerts for the ASLMO.

Table 286 – ASLMO alerts

Standard object type name(s): Application sublayer management object (ASLMO)					
Standard object type identifier: 121					
Description of the alert: Malformed APDU alert					
Alert class (Enumerated: alarm or event)	**Alert category (Enumerated: device diagnostic, comm. diagnostic, security, or process)**	**Alert type (Enumerated: based on alert category)**	**Alert priority**	**Associated data**	**Description of associated data included with alert**
Event	Communication diagnostic	malformedAPDU CommunicationAlert	7 (i.e., a mid-range medium-priority alert)	No special standard type is defined, because the protocol content does not correspond to an attribute of the ASLMO. Rather, it is constructed within the protocol as an implicit sequence, identifiable by the combination of alert class, alert category and alert type	Three elements are included in the following sequence: a) source address of malformed APDUS (IPv6Address) b) threshold value exceeded (Unsigned16) c) time interval in which threshold was exceeded (TAITimeDifference)

12.19.6 DMAP services invoked by application sublayer

If the configured ASL malformed APDUs interval and ASL malformed APDUs threshold are both non-zero, the ASL shall commence keeping malformed APDU statistics. If the threshold is reached prior to or upon expiration of the configured time interval, the ASL shall report to the DMAP that a communication diagnostic alert should be generated. The data provided by ASL to the DMAP shall include:

a) an indication that malformedAPDUsThresholdReached situation has been reached; and

NOTE This is indicated if and when the ASL malformed APDUs threshold is reached for malformed APDUs received from a single source IPv6Address within the configured ASL malformed APDUs interval.

b) diagnostic information regarding the malformed APDUs detected, such as:
   - number of APDUs received that did not have correct size,
   - number of APDUs received with an invalid service identifier, and
   - number of APDUs received with service identifier misused; and

c) the source IPv6Address of the malformed APDUs; and

d) the threshold value that was reached; and

e) the time duration over which the malformed APDUs were received.

   This time interval is calculated as the time from detection of the first malformed APDU from the indicated device by the ASL and the detection of the malformed APDU that resulted in the ASL threshold limit being reached for the indicated device. This time duration shall be less than or equal to the configured time interval established in the ASL management parameters.

12.19.7 Process industries standard objects

12.19.7.1 General

The standard objects defined in this standard are included to address the basic needs of the process industry. The unified field theory (for process control) that underlies this standard defines standard field objects by leveraging existing field device object definitions from field-proven object-oriented process control system standards. The set of objects has been selected based on commonality of use and is defined to facilitate interworkability and limited interoperability (within its domain of application) among devices.

NOTE 1 The terminology used here, including SOE, PV, OP, OOS, MAN and AUTO, is that common to the process industries.

NOTE 2 This standard presumes that any access restrictions to object attributes that are necessary to satisfy system usage requirements, are enforced by human interface devices and/or gateway devices.

To support timestamps used in process control industry alarm reports, the time value construct used to represent time shall support a coding accuracy of 1 ms. This accuracy is necessary when supporting the high speed resolution typical of the process industry SOE (sequence of events) applications.

12.19.7.2 Process industries user application objects

A basic list of user application objects is anticipated for the process control industries profile. The unified field objects (UFOs) defined in this standard are:

- analog input object,
- analog output object,
- binary input object, and
- binary output object.

Application object control mode supports the following modes:

- Target mode is the mode to which the device was commanded to transition. This may be different from the actual mode if the device cannot accept the target mode due to error, etc.
- Actual mode is the current mode of the object.
- Normal mode is the operating mode of the block that is desired by the responsible control engineer. Normal mode is one of the modes other than OOS, that is designated as “normal operation” for the block.
- Permitted modes represent the set of modes that are valid for this object. This is a filter that can be applied to limit the target mode of the block. For example, manual mode may be disabled this way. OOS is always included in the set of permitted modes.

The following modes are supported:

– OOS (out-of-service): The device is not actively measuring the PV (process variable) value or not accepting the OP (output point) value. Another common name for this mode is “inactive”. The value and associated status indicating the OOS condition are still communicated by the device. This is not intended to disable communication.

– MAN (manual): The PV can be manually entered by the operator and used for open loop control with a human in the loop or for overriding faulty measurement. MAN is also useful for testing.

– AUTO (automatic): The device is actively measuring its PV value or accepting its OP value.

A structured attribute is required to be added for each type of alert report supported by the object. This attribute supports enabling/disabling of an alert report and establishes the alert priorities.

For alarms, a structured attribute may additionally be required to establish alarm limits, to indicate if an alarm is present.

12.19.7.3 Analog input user object

12.19.7.3.1 General

A standard analog input user object representing an encapsulation of an analog input is defined. If multiple analog inputs are represented by a device, multiple analog input user objects should be instantiated. Object type-specific attributes of this object include:

- process value: a floating point value represented in engineering units and status;
- mode: a structured attribute representing target mode, actual mode, permitted mode, and normal mode;
- corresponding concentrator object: specifies the concentrator associated to publish the PV; and
- scale: represents the range and units of the process value via a structured attribute that indicates the 0 % and 100 % limits of the nominal value range, a coded representation of engineering units, and the number of significant digits that should be used for display.

Standard alerts for this object will also be defined.

12.19.7.3.2 Object attributes

An analog input object has the attributes defined in Table 287.

Table 287 – Analog input object attributes

Standard object type name: Analog input object				
Standard object type identifier: 99				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
ObjectIdentifier	Object key identifier	Unique identifier for the object	Type: Unsigned16 Classification: Constant Valid range: > 0	N/A
Reserved for future use	0	—	—	—
PV	1	Measurement variable in engineering units of the sensor	Type: Process control value and status for analog value Classification: Dynamic Accessibility: Read only Default value: NaN Status: Unknown; Substatus: Unknown Limit condition: Not limited Valid range: See definition of process control value and status for analog value structure	Analog process value and status of that value. Accessibility is read/write only when MODE.Target=MAN. See 12.19.7.2. When a write to PV is done, the device may implement this as a write to a non-network visible internal variable, and use the non-visible value when constructing the value it represents for the PV. As appropriate, the device may report a different status for the PV than that which was provided in the write request
Mode	2	Mode	Type: Process control mode Classification: Static Accessibility: Read only for actual mode; target mode, permitted mode, and normal mode all have read/write access Default value: Actual mode value indicates OOS Valid range: See Process control mode structure type definition	Actual, target, permitted, and normal mode values
Reserved for future use	3	—	—	—

Standard object type name: Analog input object				
Standard object type identifier: 99				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Scale	4	Range and units of the measurement variable	Type: Process control scaling data	Scaling information for the analog process value
			Classification: Static	
			Accessibility: Read/write	
			Default value: Engineering units values for 0 % and for 100 % BOTH indicate 0	
			Valid range: See definition of scale structure type	
Reserved for future use by this standard	5..25	—	—	*N* octets of presently undefined content

12.19.7.3.3 Standard object methods

An analog input object has the methods defined in Table 288.

Table 288 – Analog input object methods

Standard object type name: Analog input object		
Standard object type identifier: 99		
Method name	**Method ID**	**Method description**
Null	0	Reserved by standard for future use
Reserved for future use by this standard	0..127	These method identifiers are reserved for future use by this standard
Implementation-specific use	128..255	These method identifiers are available for implementation-specific use

12.19.7.3.4 Alerts

An analog input may report the alerts shown in Table 289. If an alert is supported, a corresponding alert descriptor attribute shall be added to the analog input object to describe the characteristics of the alert.

Table 289 – Analog input alerts

Standard object type name(s): Analog input						
Standard object type identifier: 99						
Description of the alert: Analog input alerts						
Alert class (Enumerated: alarm or event)	**Alert category (Enumerated: device diagnostic, comm. diagnostic, security, or process)**	**Alert type(s) (Enumerated: based on alert category)**		**Alert priority**	**Associated data: type and size**	**Description of associated data included with alert**
Alarm	Process	1	High	Any	Type: Float32	Process variable
Alarm	Process	2	HighHigh	Any	Type: Float32	Process variable
Alarm	Process	3	Low	Any	Type: Float32	Process variable
Alarm	Process	4	LowLow	Any	Type: Float32	Process variable
Alarm	Process	5	DeviationLow	Any	Type: Float32	Process variable
Alarm	Process	6	DeviationHigh	Any	Type: Float32	Process variable
Alarm	Process	0	OutOfService	Any	Type: Float32	Process variable

12.19.7.4 Analog output user object

12.19.7.4.1 General

A standard analog output user object represents an encapsulation of an analog output. If multiple analog outputs are represented by a device, multiple analog output user objects should be instantiated. Object type-specific attributes of this object include:

- commanded output value: a floating point value represented in engineering units and status;
- mode: a structured attribute representing target mode, actual mode, permitted mode, and normal mode;
- Readback: value and status of the actual position;
- provider of OP value: indicates the source of the OP value;
- corresponding concentrator object: specifies the concentrator associated to publish the Readback value; and
- scale: represents the range and units of the process value via a structured attribute that indicates the 0 % and 100 % limits of the nominal value range, a coded representation of engineering units, and the number of significant digits that should be used for display.

12.19.7.4.2 Object attributes

An analog output object has the attributes defined in Table 290.

Table 290 – Analog output attributes *(1 of 2)*

Standard object type name: Analog output object				
Standard object type identifier: 98				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
ObjectIdentifier	Object key identifier	Unique identifier for the object	Type: Unsigned16 Classification: Constant Valid range: > 0	N/A
Reserved for future use	0	—	—	—
OP	1	Output value for the actuator	Type: Process control value and status for analog value structure Classification: Dynamic Accessibility: Read/write Default value: NaN Status: Unknown Substatus: Unknown Limit condition: Not limited Valid range: See definition of process control value and status for analog value structure	This is the standard attribute that serves as the destination attribute for a publication from another object
Mode	2	Mode	Type: Process control mode Classification: Static Accessibility: Read only for actual mode; target mode, permitted mode, and normal mode all have read/write access Default value: Actual mode value indicates OOS Valid range: See Process control mode structure type definition	Actual, target, permitted, and normal mode values
Reserved for future use	3	—	—	—
Readback	4	Readback value – the actual position of the OP	Type: Process control value and status for analog value Classification: Dynamic Accessibility: Read only Default value: NaN; Status: Unknown Substatus: Unknown Limit condition: Not limited Valid range: See definition of process control value and status for analog value structure	Analog process value and status of that value. This is the standard attribute that is published from this object. If this object is extended, such that another attribute needs to be published, (a) concentrator object(s) represent(s) the resulting publication(s)

Table 290 *(2 of 2)*

Standard object type name: Analog output object				
Standard object type identifier: 98				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Reserved for future use	5	—	—	—
Scale	6	Range and units of the output variable	Type: Process control scaling data structure	Scaling information
			Classification: Static	
			Accessibility: Read/write	
			Default value: Engineering units values for 0 % and for 100 % BOTH indicate 0	
			Valid range: See definition of Scale structure type	
Reserved for future use by this standard	7..25	—	—	*N* octets of presently undefined content

12.19.7.4.3 Standard object methods

An analog output object has the methods defined in Table 291.

Table 291 – Analog output object methods

Standard object type name: Analog output object		
Standard object type identifier: 98		
Method name	**Method ID**	**Method description**
Null	0	Reserved by this standard for future use
Reserved for future use by this standard	0..127	These method identifiers are reserved for future use by this standard
Implementation-specific use	128..255	These method identifiers are available for implementation-specific use

12.19.7.4.4 Alerts

An analog output may report the alerts shown in Table 292. If an alert is supported, a corresponding alert descriptor attribute shall be added to the analog output object to describe the characteristics of the alert.

Table 292 – Analog output alerts

Standard object type name(s): Analog output						
Standard object type identifier: 98						
Description of the alert: Analog output alerts						
Alert class (Enumerated: alarm or event)	**Alert category (Enumerated: device diagnostic, comm. diagnostic, security, or process)**	**Alert type(s) (Enumerated: based on alert category)**		**Alert priority**	**Associated data: type and size**	**Description of associated data included with alert**
Alarm	Process	1	High	Any	Type: Float32	Process variable
Alarm	Process	2	HighHigh	Any	Type: Float32	Process variable
Alarm	Process	3	Low	Any	Type: Float32	Process variable
Alarm	Process	4	LowLow	Any	Type: Float32	Process variable
Alarm	Process	5	DeviationLow	Any	Type: Float32	Process variable
Alarm	Process	6	DeviationHigh	Any	Type: Float32	Process variable
Alarm	Process	0	OutOfService	Any	Type: Float32	Process variable

12.19.7.5 Binary input user object

12.19.7.5.1 General

A standard binary input user object represents an encapsulation of a single binary input. If multiple binary inputs are represented by a device, multiple binary input user objects should exist to represent them. Object type specific attributes of this object include:

- process value: binary value and status;
- mode: a structured attribute representing target mode, actual mode, permitted mode, and normal mode; and
- corresponding concentrator object: specifies the concentrator associated to publish the process value.

12.19.7.5.2 Object attributes

A binary input object has the attributes defined in Table 293.

Table 293 – Binary input object attributes

Standard object type name: Binary input object				
Standard object type identifier: 97				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
ObjectIdentifier	Object key identifier	Unique identifier for the object	Type: Unsigned16 Classification: Constant Valid range: > 0	N/A
Reserved for future use	0	—	—	—
PV_B	1	Discrete measurement variable	Type: Process control value and status for discrete value Classification: Dynamic Accessibility: Read only Default value: 0 Status: Unknown Substatus: Unknown Limit condition: Not limited Valid range: See definition of process control value and status for discrete value structure	Binary process value and status of that value. This is the standard attribute that is published from this object. If this object is extended, such that another attribute needs to be published, (a) concentrator object(s) represent(s) the resulting publication(s). Accessibility is read/write only when in MAN mode. See 12.19.7.2. When a write to PV_B is done, the device may implement this as a write to a non-network visible internal variable, and use the non-visible value when constructing the value it represents for the PV_B. As appropriate, the device may report a different status for the PV_B than that which was provided in the write request
Mode	2	Mode	Type: Process control mode Classification: Static Accessibility: Read only for actual mode; target mode, permitted mode, and normal mode all have read/write access Default value: Actual mode value indicates OOS Valid range: See Process control mode structure type definition	Actual, target, permitted, and normal mode values
Reserved for future use by this standard	3..25	—	—	*N* octets of presently undefined content

12.19.7.5.3 Standard object methods

A binary input object has the methods defined in Table 294.

Table 294 – Binary input object methods

Standard object type name: Binary input object		
Standard object type identifier: 97		
Method name	**Method ID**	**Method description**
Null	0	Reserved by this standard for future use
Reserved for future use by this standard	0..127	These method identifiers are reserved for future use by this standard
Implementation-specific	128..255	These method identifiers are available for implementation-specific use

12.19.7.5.4 Alerts

An analog output may report the alerts shown in Table 295. If an alert is supported, a corresponding alert descriptor attribute shall be added to the binary input object to describe the characteristics of the alert.

Table 295 – Binary input alerts

Standard object type name(s): Binary input object						
Standard object type identifier: 97						
Description of the alert: Binary input alerts						
Alert class (Enumerated: alarm or event)	**Alert category (Enumerated: device diagnostic, comm. diagnostic, security, or process)**	**Alert type(s) (Enumerated: based on alert category)**		**Alert priority**	**Associated data: type and size**	**Description of associated data included with alert**
Alarm	Process	1	DiscreteAlarm	Any	Type: Boolean	Process value
Alarm	Process	0	OutOfService	Any	Type: Boolean	Process value

12.19.7.6 Binary output user object

12.19.7.6.1 General

A standard binary output user object represents an encapsulation of a single binary output. If multiple binary outputs are represented by a device, multiple binary output user objects should exist to represent them. Object type specific attributes of this object include:

- commanded output value: binary value and status;
- mode: a structured attribute representing target mode, actual mode, permitted mode, and normal mode;
- provider of OP value: indicates the source of the OP value;
- Readback value: binary value and status; and
- corresponding concentrator object: specifies the concentrator associated to publish the Readback_B value.

12.19.7.6.2 Object attributes

A binary output object has the attributes defined in Table 296.

Table 296 – Binary output attributes

Standard object type name: Binary output object				
Standard object type identifier: 96				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
ObjectIdentifier	Object key identifier	Unique identifier for the object	Type: Unsigned16 Classification: Constant Valid range: > 0	N/A
Reserved for future use	0	—	—	—
OP_B	1	Discrete measurement variable	Type: Process control value and status for discrete value structure Classification: Dynamic Accessibility: Read/write Default value: 0 Status: Unknown Substatus: Unknown Limit condition: Not limited Valid range: See definition of process control value and status for discrete value structure	This is the standard attribute that is the destination for a publication from another object
Mode	2	Mode	Type: Process control mode Classification: Static Accessibility: Read only for actual mode; target mode, permitted mode, and normal mode all have read/write access Default value: Actual mode value indicates OOS Valid range: See Process control mode structure type definition	Actual, target, permitted, and normal mode values
Reserved for future use	3	—	—	—
Readback_B	4	Readback value of actual position of the actuator	Type: Process control value and status for discrete value Classification: Dynamic Accessibility: Read only Default value: 0; Status: Unknown Substatus: Unknown Limit condition: Not limited Valid range: See definition of process control value and status for analog value structure	Analog process value and status of that value. This is the standard attribute that is published from this object. If this object is extended, such that another attribute needs to be published, a concentrator object(s) represents the resulting publication(s)
Reserved for future use by this standard	5..25	—	—	*N* octets of presently undefined content

12.19.7.6.3 Standard object methods

A binary output object has the methods defined in Table 297.

Table 297 – Binary output object methods

Standard object type name: Binary output object		
Standard object type identifier: 96		
Method name	**Method ID**	**Method description**
Null	0	Reserved by standard for future use
Reserved for future use by this standard	0..127	These method identifiers are reserved for future use by this standard
Implementation-specific use	128..255	These method identifiers are available for implementation-specific use

12.19.7.6.4 Alerts

A binary output may report the alerts shown in Table 298. If an alert is supported, a corresponding alert descriptor attribute shall be added to the binary output object to describe the characteristics of the alert.

Table 298 – Binary output alerts

Standard object type name(s): Binary output object						
Standard object type identifier: 96						
Description of the alert: Binary output alerts						
Alert class (Enumerated: alarm or event)	**Alert category (Enumerated: device diagnostic, comm. diagnostic, security, or process)**	**Alert type(s) (Enumerated: based on alert category)**		**Alert priority**	**Associated data: type and size**	**Description of associated data included with alert**
Alarm	Process	1	DiscreteAlarm	Any	Type: Boolean	Process value
Alarm	Process	0	OutOfService	Any	Type: Boolean	Process value

12.19.8 Factory automation industries profile

12.19.8.1 General

Additional standard objects to support the needs of factory automation may be added in future releases of this standard. More detailed thought specific to factory automation is deferred to a later release of this standard's standardization activity. Examples of objects that may be defined to meet the needs of factory automation may include:

a) binary input user object (e.g., a contact);

b) binary output user object (e.g., a coil);

c) analog input user object;

d) analog output user object;

e) register input user object (e.g., to map 16 bits of two-state information often found in PLC input registers);

f) register output user object (e.g., to map 16 bits of two-state information often found in PLC output registers);

g) multi-state input user object (e.g., to map 8 bits of state information for a valve with enumerated states such as opening, open, closing, closed, or to map a unidirectional motor with states such as off, starting, running, stopping); and

h) multi-state output user object (e.g., to map 8 bits of state output information for an output device with enumerated states).

Standard alerts for these objects may also be defined.

12.19.8.2 Manufacturer-specific objects

Vendors may also define vendor- or device-specific custom objects. An example of a vendor-specific object for process systems is an equipment-mounted display object, in which a string can be stored for display to a human near the device.

12.20 Process control industry standard data structures

12.20.1 General

The process control industry standard data structures used shall be the data structures conveyed by the protocol defined by this standard. Industry-independent standard data structures and process control industry data structures are both summarized in Annex L.

NOTE Vendor-specific data structure definitions are not supported.

12.20.2 Status for analog information

The status for analog information is a packed fixed format octet containing the items shown in Table 299.

Table 299 – Status octet

Bit 7 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)
Quality		**<reserved>**	**Quality dependent substatus**			**Limit condition**	
0 = bad (value should not be used)		This bit shall always be set to zero	Named values when quality=bad: 0: non-specific; 1: configuration error; 2: not connected; 3: device failure; 4: sensor failure; 5: no communication with a lastUsableValue; 6: no communication without a lastUsableValue; 7: out of service (value is not being computed)			Named values: 0 = not limited; 1 = low limit 2 = high limit; 3 = constant (both high and low limited)	
1 = uncertain (value of less than normal quality)			Named values when quality=uncertain: 0: non-specific; 1: last usable value; 2: substituted or manual entry; 3: initial value; 4: sensor conversion inaccurate; 5: range limits exceeded; 6: sub normal; 7: reserved				
2 = good (quality of value is good, but an alarm condition may exist)			Named values when quality=good: 0: no special conditions exist; 1..7: reserved				
3 = reserved			All values are reserved. Within this standard, this shall always be set to zero				
NOTE The definitions for the status octet are a subset of those defined by the HART Communication Foundation, the Fieldbus Foundation, and the OPC Foundation.							

12.20.3 Value and status for analog information

As status does not indicate substitution, network-initiated writes using the analog data type structures are not permitted by this standard.

The structure of analog information is shown in Table 300.

Table 300 – Data type: Process control value and status for analog value

Standard data type name: Process control value and status for analog value		
Standard data type code: 65		
Element name	**Element identifier**	**Element scalar type**
Status	1	Type: BitString8 Classification: Dynamic Valid value set: See Table 299
Value	2	Type: Float32 Classification: Dynamic

12.20.4 Value and status for binary information

As status does not indicate substitution, network-initiated writes to digital data type structures are not permitted.

The structure of digital information is shown in Table 301.

Table 301 – Data type: Process control value and status for binary value

Standard data type name: Process control value and status for binary value		
Standard data type code: 66		
Element name	**Element identifier**	**Element scalar type**
Status	1	Type: BitString8 Accessibility: May vary by use Valid value set: See Table 299
Value	2	Type: Boolean

12.20.5 Process control mode

Elements in process control mode are shown in Table 302.

Table 302 – Data type: Process control mode

Standard data type name: Process control mode		
Standard data type code: 69		
Element name	**Element identifier**	**Element scalar type**
Target	1	Type: BitString8 Classification: Static Accessibility: Read/write Default value: OOS Valid value set : See Table 303
Actual	2	Type: BitString8 Classification: Dynamic Accessibility: Read only Default value: OOS Valid value set : See Table 303
Permitted	3	Type: BitString8 Classification: Static Accessibility: Read/write Default value: OOS Valid value set : See Table 303
Normal	4	Type: BitString8 Classification: Static Accessibility: Read/write Default value: OOS Valid value set : See Table 303

The value of each element of the mode structure is represented by a bitstring containing the bits in Table 303, where the <reserved> bits shall be set to zero (0).

Table 303 – Data type: Process control mode bitstring

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<reserved>	<reserved>	<reserved>	AUTO	MAN	<reserved>	<reserved>	OOS

That is:

- OOS-only is equal to 0x01, with the equivalent decimal value of 1.
- MAN-only is equal to 0x08, with the equivalent decimal value of 8.
- AUTO-only is equal to 0x10, with the equivalent decimal value of 16.

12.20.6 Scaling

Elements in process control scaling are shown in Table 304.

Table 304 – Data type: Process control scaling

Standard data type name: Process control scaling		
Standard data type code: 68		
Element name	**Element identifier**	**Element scalar type**
Engineering units at 100 % (the upper range of the associated object parameter)	1	Type: Float32 Classification: Static Accessibility: Read/write Default value : 0
Engineering units at 0 % (the lower range of the associated object parameter)	2	Type: Float32 Classification: Static Accessibility: Read/write Default value : 0
Units index for both SI units and traditional (non-SI) units [a] – range 1 000..1 999 – other codes reserved	3	Type: Unsigned16 Classification: Static Accessibility: Read/write
Location of the decimal point/decimal sign (represents the number of digits to the right of the decimal point/decimal sign, i.e., the significance of the fractional part of the associated value)	4	Type: Unsigned8 Classification: Static Accessibility: Read/write Default value : 0

[a] As specified in *Standard Units Codes Table*, published in ISA *Handbook of Measurement Equations and Tables*, 2nd Edition, ISBN 978 1 55617 946 4. This index is also published by the Fieldbus Foundation as TN-016:2007, Table 3.1.

12.21 Additional tables

12.21.1 Process control profile standard objects

Table 305 lists process control profile standard objects.

Table 305 – Process control standard objects

Object type	Defined by	Standard object type identifier
Analog input	12.19.7.3	99
Analog output	12.19.7.4	98
Binary input	12.19.7.5	97
Binary output	12.19.7.6	96

12.21.2 Services

Table 306 provides a list of services.

Table 306 – Services

Service	Use
Read	Read the value of one or more attributes from one or more objects of a UAP
Write	Write the value to one or more attributes of one or more objects of a UAP
Execute	Execute a set of functions on object instances of a UAP
Publish	Periodically publish data to subscribers
AlertReport	Report one or more alert conditions
AlertAcknowledge	Acknowledge an AlertReport
Tunnel	Pass payload through

12.22 Coding

12.22.1 General

The conditions for encoding wireless APDUs in this standard include the following considerations:

- Some messages occur often, such as periodic publications.
- Messages should be short, to preserve battery power.
- There should be minimal selection of ASL service types.

The maximum size of an APDU (which is a TSDU) is determined by subtracting (the size of the information TL adds to the TSDU to form the TPDU) from (the Assigned_Max_NSDU_Size), where Assigned_Max_NSDU_Size is an output parameter of the method used to establish a communication contract. That is:

maxAPDUSize = Assigned_Max_NSDU_Size – sizeOF(TLInfoAddedtoAPDU)

See 6.3.11.2.5 for further details of communication contract establishment.

ASL coding shall ensure that the maximum APDU size is not exceeded.

NOTE IETF RFC 2348 provides recommendations on the maximum size of NPDUs.

12.22.2 Coding rules for application protocol data units

12.22.2.1 General

The coding rules defined in 12.22.2 represent bit 0 as the least significant bit (LSB) in the value represented.

All APDUs contain an AL header, and a service type-specific APDU content. Table 307 indicates the general coding construct of an APDU.

Table 307 – Application messaging format

May be 1, 2, 3, or 5 octets (see 12.22.2.3)	*N* octets
ASDU header (ensures routing to correct objects; provides service type identification)	Service-specific content

12.22.2.2 Concatenation

APDUs can be concatenated together, and the concatenation of individual APDUs may be given to the TL as a single TSDU. The size of this TSDU shall not exceed the maximum APDU size for communications relative to the corresponding communication contract between the sending and receiving applications.

Table 308 describes the format of concatenated APDUs in a single TSDU.

Table 308 – Concatenated APDUs in a single TSDU

APDU_1	...	APDU_n

Concatenation can be used to ensure that when one of the concatenated APDUs is received, all have been received, thus providing a basis for multi-component actions that are atomic with respect to inter-device communications.

NOTE How the ASL determines when and what to concatenate is a local matter.

12.22.2.3 AL header

The AL header supports four object identifier addressing modes which determine header construction beyond the first octet. The object identifier addressing modes are:

- four-bit object identifier addressing mode;
- eight-bit (1 octet) object identifier addressing mode;
- sixteen-bit (2 octets) object identifier addressing mode; and
- inferred addressing mode, which may be used only in the second and subsequent APDUs of a TSDU that contains multiple concatenated APDUs.

Identification of the UAP containing the object is a function of the TL.

The object identifier addressing mode in use for APDU interpretation is indicated in bits 5 and 6 of the first octet of the APDU header. Table 309 represents the coding of this first APDU header octet.

Table 309 – Object addressing

Octets	Bits							
	7	6	5	4	3	2	1	0
1	Service primitive type (.req = 0 .resp = 1)	Object identifier addressing mode Named values: 00: 4-bit mode 01: 8-bit mode 10: 16-bit mode 11: inferred mode		ASL service type				

12.22.2.4 Object identifier coding

12.22.2.4.1 General

In all header constructions, the source object identifier represents the object identifier in the application that is initiating the ASL service primitive (that is, the initiator of a .request or a .response primitive), and the destination object identifier represents the object identifier in the application that is receiving the ASL service primitive (that is, the recipient of an .indication or a .confirmation primitive).

12.22.2.4.2 Four-bit object identifier addressing mode

A four-bit object identifier addressing mode indicates that the source object identifier and the destination object identifier each can be expressed in a 4-bit unsigned integer. This addressing mode provides for optimal header compaction for application processes with a small number of objects. This mode is described in Table 310.

Table 310 – Four-bit addressing mode APDU header construction

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Service primitive type	0	0	ASL service type				
1	Source object identifier				Destination object identifier			

12.22.2.4.3 Eight-bit object identifier addressing mode

An eight-bit object identifier addressing mode indicates that the source object identifier and the destination object identifier each can be expressed in an 8-bit unsigned integer, and further that at least one of the object identifiers cannot be expressed in a 4-bit unsigned integer. This mode is described in Table 311.

Table 311 – Eight-bit addressing mode APDU header construction

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Service primitive type	0	1	ASL service type				
1	Source object identifier							
1	Destination object identifier							

12.22.2.4.4 Sixteen-bit object identifier addressing mode

A sixteen-bit object identifier addressing mode indicates that the source object identifier and the destination object identifier each can be expressed in a 16-bit unsigned integer, and further that at least one of the object identifiers cannot be expressed in an 8-bit unsigned integer. This mode is described in Table 312.

Table 312 – Sixteen-bit addressing mode APDU header construction

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Service primitive type	1	0	ASL service type				
2	Source object identifier							
2	Destination object identifier							

12.22.2.4.5 Inferred object identifier addressing mode for optimized concatenations

An inferred object identifier addressing mode is an optimization technique used only within a concatenated APDU. The intent of this technique is to save octets transmitted by eliminating redundant source and object identifiers, which can be determined from the most recently parsed APDU contained within the same APDU concatenation.

Inferred object addressing shall not be indicated in the first APDU of a concatenation.

NOTE Any APDU indicating an inferred object addressing mode in the first APDU met in ASL parsing is considered a malformed APDU.

An example is included in Table 313.

Table 313 – Inferred addressing use case example

APDU_1	APDU_2	APDU_3	APDU_4	APDU_5
00 object identifier addressing mode	11 object identifier addressing mode (indicates use source and destination OIDs from APDU_1)	11 object identifier addressing mode (indicates use source and destination OIDs from APDU_2, which is used the source and destination OIDs from APDU_1)	01 object identifier addressing mode	11 object identifier addressing mode (indicates use source and destination OIDs from APDU_4)
APDU_1 includes: – 00 addressing mode; – service type; – 4-bit source object identifier; – 4-bit destination object identifier; service-specific payload	APDU_2 includes: – 11 addressing mode; – service type; – service-specific payload	APDU_3 includes: – 11 addressing mode; – service type; – service-specific payload	APDU_4 includes: – 01 addressing mode – service type – 8-bit source object identifier; – 8-bit destination object identifier; service-specific payload	APDU_5 includes: – 11 addressing mode; – service type; – service-specific payload

Table 314 describes the construction of the inferred addressing mode APDU header.

Table 314 – Inferred addressing mode APDU header construction

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Service primitive type	1	1	ASL service type				

12.22.2.5 Object attribute coding

12.22.2.5.1 General

Object attribute coding is determined by an attribute identifier format. The format may indicate:

- Six-bit, not indexed: The attribute fits into an Unsigned6 integer, and is not indexed.
- Six-bit, singly indexed: The attribute fits into an Unsigned6 integer, and requires one index. The attribute index is extensible, as indicated by the first bit of the index. If the first bit of the index is 0, the index is 7 bits in size. If the first bit of the index is 1, the index is 15 bits in size.
- Six-bit, doubly indexed: The attribute fits into an Unsigned6 integer, and requires two indices. The attribute indices are individually extensible; that is, the first index may be 7 bits or 15 bits in size, and the second index also may be either 7 bits or 15 bits in size. The size of the index is determined by the first bit of the index. If the first bit of the index is 0, the index is 7 bits in size. If the first bit of the index is 1, the index is 15 bits in size.
- Twelve-bit, not indexed: The attribute fits does not fit into an Unsigned12 integer. The attribute is not indexed.
- Twelve-bit, singly indexed: The attribute fits into an Unsigned12 integer, and requires one index. The attribute index is extensible, as indicated by the first bit of the index. If

the first bit of the index is 0, the index is 7 bits in size. If the first bit of the index is 1, the index is 15 bits in size.

- Twelve-bit, doubly indexed: The attribute fits into an Unsigned12 integer, and requires two indices. The attribute indices are individually extensible; that is, the first index may be 7 bits or 15 bits in size, and the second index also may be either 7 bits or 15 bits in size. The size of the index is determined by the first bit of the index. If the first bit of the index is 0, the index is 7 bits in size. If the first bit of the index is 1, the index is 15 bits in size.

NOTE Refer to 12.23.1.3 for the definitions of Unsigned6 and Unsigned12.

12.22.2.5.2 Six-bit attribute identifier, not indexed

Table 315 indicates the coding for a six-bit attribute identifier that is not an indexed or structured attribute.

Table 315 – Six-bit attribute identifier, not indexed

Number of octets	Number of octets							
	7	6	5	4	3	2	1	0
1	Attribute short form (value = binary 00)		Attribute identifier					

12.22.2.5.3 Six-bit attribute identifier, singly indexed forms

Table 316 and Table 317 indicate the coding for a six-bit attribute identifier that may be accessed using a single index.

Table 316 – Six-bit attribute identifier, singly indexed, with 7-bit index

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Attribute short form (value = binary 01)		Attribute identifier					
1	0	Index						

Table 317 – Six-bit attribute identifier, singly indexed, with 15-bit index

Number of octets	bits							
	7	6	5	4	3	2	1	0
1	Attribute short form (value = binary 01)		Attribute identifier					
2	1	Index (high-order 7 bits)						
	Index (low-order 8 bits)							

12.22.2.5.4 Six-bit attribute identifier, doubly indexed forms

Table 318, Table 319, Table 320, and Table 321 indicate the coding for a six-bit attribute identifier that may be accessed using two indices.

Table 318 – Six-bit attribute identifier, doubly indexed, with two 7-bit indices

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Attribute short form (value = binary 10)		Attribute identifier					
1	0	Index 1						
1	0	Index 2						

Table 319 – Six-bit attribute identifier, doubly indexed, with two 15-bit indices

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Attribute short form (value = binary 10)		Attribute identifier					
2	1	Index 1 (high-order 7 bits)						
	Index 1 (low-order 8 bits)							
2	1	Index 2 (high-order 7 bits)						
	Index 2 (low-order 8 bits)							

Table 320 – Six-bit attribute identifier, doubly indexed, with first index seven bits long and second index fifteen bits long

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Attribute short form (value = binary 10)		Attribute identifier					
1	0	Index 1						
2	1	Index 2 (high-order 7 bits)						
	Index 2 (low-order 8 bits)							

Table 321 – Six-bit attribute bit attribute identifier, doubly indexed, with first index fifteen bits long and second index seven bits long

Number of octets	bits							
	7	6	5	4	3	2	1	0
1	Attribute short form (value = binary 10)		Attribute identifier					
2	1	Index 1 (high-order 7 bits)						
	Index 1 (low-order 8 bits)							
1	0	Index 2						

12.22.2.5.5 Twelve-bit attribute identifier, not indexed

Table 322 indicates the coding for a twelve-bit attribute identifier that is not indexed.

Table 322 – Twelve-bit attribute identifier, not indexed

Number of octets	Bits							
	7	6	5	4	3	2	1	0
2	Attribute short form (value = binary 11)		Attribute long form, index form = binary 00		Attribute identifier (high-order 4 bits)			
	Attribute identifier (low-order 8 bits)							

12.22.2.5.6 Twelve-bit attribute identifier, singly indexed coding forms

Table 323 and Table 324 indicate the coding for a twelve-bit attribute identifier that is accessed using a single index.

Table 323 – Twelve-bit attribute identifier, singly indexed with 7-bit index

Number of octets	Bits							
	7	6	5	4	3	2	1	0
2	Attribute short form (value = binary 11)		Attribute long form, index form = binary 01		Attribute identifier (high-order 4 bits)			
	Attribute identifier (low-order 8 bits)							
1	0	Index						

Table 324 – Twelve-bit attribute identifier, singly indexed with 15-bit index

Number of octets	Bits							
	7	6	5	4	3	2	1	0
2	Attribute short form (value = binary 11)		Attribute long form, index form = binary 01		Attribute identifier (high-order 4 bits)			
	Attribute identifier (low-order 8 bits)							
2	1	Index 1 (high-order 7 bits)						
	Index 1 (low-order 8 bits)							

12.22.2.5.7 Twelve-bit attribute identifier, doubly indexed coding forms

Table 325, Table 326, Table 327, and Table 328 indicate the coding for a twelve-bit attribute identifier that is accessed using two indices.

Table 325 – Twelve-bit attribute identifier, doubly indexed with two 7-bit indices

Number of octets	Bits							
	7	6	5	4	3	2	1	0
2	Attribute long form (value = binary 11)		Attribute long form, index form = binary 10		Attribute identifier (high-order 4 bits)			
	Attribute identifier (low-order 8 bits)							
1	0	Index 1						
1	0	Index 2						

Table 326 – Twelve-bit attribute identifier, doubly indexed with two 15-bit indices

Number of octets	Bits							
	7	6	5	4	3	2	1	0
2	Attribute short form (value = binary 11)		Attribute long form, index form = binary 10		Attribute identifier (high-order 4 bits)			
	Attribute identifier (low-order 8 bits)							
2	1	Index 1 (high-order 7 bits)						
	Index 1 (low-order 8 bits)							
2	1	Index 2 (high-order 7 bits)						
	Index 2 (low-order 8 bits)							

Table 327 – Twelve-bit attribute identifier, doubly indexed with first index 7 bits long and second index 15 bits long

Number of octets	Bits							
	7	6	5	4	3	2	1	0
2	Attribute short form (value = binary 11)		Attribute long form, index form = binary 10		Attribute identifier (high-order 4 bits)			
	Attribute identifier (low-order 8 bits)							
1	0	Index 1						
2	1	Index 2 (high-order 7 bits)						
	Index 2 (low-order 8 bits)							

Table 328 – Twelve-bit attribute identifier, doubly indexed with the first index 15 bits long and the second index 7 bits long

Number of octets	Bits							
	7	6	5	4	3	2	1	0
2	Attribute short form (value = binary 11)		Attribute long form, index form = binary 10		Attribute identifier (high-order 4 bits)			
	1	Index 1 (high-order 7 bits)						
2	Index 2 (low-order 8 bits)							
	0	Index 2						

12.22.2.5.8 Reserved for future use

Table 329 identifies an attribute identifier form that is reserved for future use.

Table 329 – Twelve-bit attribute identifier, reserved form

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Attribute short form (value = binary 11)		Attribute long form, index form (reserved): binary 11		Reserved for future use			

12.22.2.6 Read

Table 330 provides coding rules for the service specific portion of a read service request APDU.

Application Request ID is an identifier that enables the client to match a service response with the original service request. A service response shall include a copy of the Request ID from the corresponding service request.

Table 330 – Coding rules for read service request

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Request ID							
...	Attribute identifier (see coding rules for attribute identifier)							

Table 331 provides coding rules for a read service response with 7-bit size field.

Table 331 – Coding rules for read service response with 7-bit size field

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Request ID							
1	Reserved for future use by this standard. For compliance with this version of this standard, these bits shall be set to 0							Forward explicit congestion control echo
1	ServiceFeedbackCode							
1	0	S=Size – conditionally included only when ServiceFeedbackCode indicates success						
S	Value – conditionally present only when ServiceFeedbackCode only if indicates success							

NOTE Refer to 12.23.3 for the definitions of ServiceFeedbackCode for AL services.

Table 332 provides coding rules for a read service response with 15-bit size field.

Table 332 – Coding rules for read service response with 15-bit size field

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Request ID							
1	Reserved for future use by this standard. For compliance with this version of this standard, these bits shall be set to 0							Forward explicit congestion control echo
1	ServiceFeedbackCode							
2	1	S[14..8]=Size – high-order 7 bits, conditionally present only when ServiceFeedbackCode indicates success						
	S[7..0]=Size – low-order 8 bits, conditionally present only when ServiceFeedbackCode indicates success							
S	Value – conditionally present only when ServiceFeedbackCode indicates success							

12.22.2.7 Write

Table 333 and Table 334 provide coding rules for a write service request.

Application Request ID is an identifier that enables the client to match a service response with the original service request. A service response shall include a copy of the Request ID from the corresponding service request.

Table 333 – Coding rules for write service request with 7-bit size field

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Request ID							
...	Attribute identifier (see attribute encoding rules)							
...	0	S=Size						
S	Value							

Table 334 – Coding rules for write service request with 15-bit size field

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Request ID							
...	Attribute identifier (see attribute encoding rules)							
2	1	S[14..8]==Size (high-order 7 bits)						
	S[7..0]=Size (low-order 8 bits)							
S	Value							

Table 335 provides coding rules for a write service response.

Table 335 – Coding rules for write service response

Number of octets	bits							
	7	6	5	4	3	2	1	0
1	Request ID							
1	Reserved for future use by this standard. For compliance with this version of this standard, these bits shall be set to 0							Forward explicit congestion control echo
1	ServiceFeedbackCode							

12.22.2.8 Execute

Table 336 and Table 337 provide coding rules for an execute service request.

Application Request ID is an identifier that enables the client to match a service response with the original service request. A service response shall include a copy of the Request ID from the corresponding service request.

Table 336 – Coding rules for execute service request with 7-bit size field

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Request identifier							
1	Method identifier							
1	0	S=Size in octets of request parameters						
S	Request parameters							

Table 337 – Coding rules for execute service request with 15-bit size field

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Request identifier							
1	Method identifier							
2	1	S[14..8]=Size in octets of response parameters (high-order 7 bits)						
	S[7..0]=Size (low-order 8 bits)							
S	Response parameters							

Table 338 and Table 339 provide coding rules for an execute service response.

Table 338 – Coding rules for execute service response with 7-bit size field

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Request identifier							
1	Reserved for future use by this standard. For compliance with this version of this standard, these bits shall be set to 0							Forward explicit congestion control echo
1	ServiceFeedbackCode							
1	0	S=Size in octets of response parameters						
S	Response parameters							

Table 339 – Coding rules for execute service response with 15-bit size field

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Request identifier							
2	Reserved for future use by this standard. For compliance with this version of this standard, these bits shall be set to 0							Forward explicit congestion control echo
1	ServiceFeedbackCode							
2	1	S[14..8]=Size in octets of response parameters (high-order 7 bits)						
	S[7..0]=Size (low-order 8 bits)							
S	Response parameters							

12.22.2.9 Tunnel

Table 340 and Table 341 provide coding rules for a tunnel service request.

Table 340 – Coding rules for tunnel service request with 7-bit size field

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	0	S=7-bit size						
S	Payload							

Table 341 – Coding rules for tunnel service request with 15-bit size field

Number of octets	Bits							
	7	6	5	4	3	2	1	0
2	1	S[14..8]=Size in octets of response parameters (high-order 7 bits)						
	S[7..0]=Size (low-order 8 bits)							
S	Payload							

Table 342 and Table 343 provide coding rules for a tunnel service response.

Table 342 – Coding rules for tunnel service response with 7-bit size field

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Reserved for future use by this standard. For compliance with this version of this standard, these bits shall be set to 0							Forward explicit congestion control echo
1	0	S=Size						
S	Payload							

Table 343 – Coding rules for tunnel service response with 15-bit size field

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Reserved for future use by this standard. For compliance with this version of this standard, these bits shall be set to 0							Forward explicit congestion control echo
2	1	S[14..8]=Size in octets of response parameters (high-order 7 bits)						
	S[7..0]=Size (low-order 8 bits)							
S	Payload							

12.22.2.10 AlertReport

Table 344 and Table 345 provide coding rules for an AlertReport service request.

Table 344 – Coding rules for AlertReport service with 7-bit associated-data size field

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Alert report ID							
2	Detecting object application process identifier (T-port)							
2	Detecting object identifier							
6	TAINetworkTime							
1	Class	Direction	Category		Alert Priority			
1	Type							
1	0	S=Size of associated data						
S	Associated data							

Table 345 – Coding rules for AlertReport service with 15-bit associated-data size field

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Alert report ID							
2	Detecting object application process identifier (T-port)							
2	Detecting object identifier							
6	TAINetworkTime							
1	Class	Direction	Category		Alert Priority			
1	Type							
2	1	S[14..8]=Size in octets of response parameters (high-order 7 bits)						
	S[7..0]=Size (low-order 8 bits)							
S	Associated data							

12.22.2.11 AlertAcknowledge

Table 346 provides coding rules for an AlertAcknowledge service request.

Table 346 – Coding rules for AlertAcknowledge service

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Alert report ID							

12.22.2.12 Publish

Table 347 provides coding rules for a native publish service.

When used in conjunction with a concentrator object, “Data” in the payload comprises the entire data communicated, which is a configured sequence of process control variables. The process control variables include both status information and process values. The structure of the data is indicated by the publishing content version. The freshness sequence number is within the scope of a particular concentrator object.

Table 347 – Coding rules for publish service for a native sequence of values

Number of octets	Bits							
	7	6	5	4	3	2	1	0
1	Publishing content version							
1	Freshness sequence number							
S	Data							

Table 348 provides coding rules for a publish service used to convey either an internally encoded octet string, or non-native data. Use of this service for non-native data enables support for tunneling.

Table 348 – Coding rules for publish service – non-native (for tunnel support)

Number of octets	Bits							
	7	6	5	4	3	2	1	0
S	Data							

The coding rules for uninterpreted varying-size data, given in Table 351, apply to a published healthReport (see 12.23.1.6).

12.22.2.13 Concatenation

Table 349 provides coding rules for a constructing a single TSDU which contains multiple logical APDUs.

Table 349 – Coding rules for concatenate service

Number of octets	Bits							
	7	6	5	4	3	2	1	0
S	SEQUENCE OF APDUs							

12.22.3 Coding of application data

12.22.3.1 General

Coding of single application data elements is always primitive. In the tables of 12.22.3, octet 1 represents the most significant octet, bit 7 represents the most significant bit within an octet, and bit 0 represents the least significant bit within an octet.

The semantics of user data are known by:

- prior agreement (e.g., tunnel payload content);
- position in the APDU with fixed field size for content; or
- existing fields in the APDU.

In these situations, no additional decoding information is added to the APDU.

Coding rules for application data are provided in Table 350 and Table 351. If the size is fixed, such as for data type OctetStringN for a given fixed value of N, then size information is implicit in the declaration, so is not explicitly conveyed in the APDU, as shown in Table 350.

Table 350 – General coding rule for size-invariant application data

Data (fixed size)

In contrast, if the size may vary, such as for data type OctetString (and not OctetStringN for any N), then the size of the actual field is explicitly conveyed in the APDU. Often that is done by prefixing the data with the size, as shown in Table 351. In other cases, the size field either is found directly in, or is computable from, some earlier-parsed field in the APDU.

Table 351 – General coding rule for size-varying application data of 0..255 octets

Unsigned8 N, size of data (in octets)	Data (size N octets)

Subclauses 12.22.3.2 through 12.22.3.8 define the data coding for standard data types.

12.22.3.2 Boolean values

NOTE The type name honors the logician George Boole, hence its capitalization.

12.22.3.2.1 Coding of Boolean values

Booleans are coded as zero/non-zero values in either a 1-bit, for packed data structures, or an 8-bit field, for relatively unpacked data structures.

12.22.3.2.2 Boolean8

The coding of a Boolean8, which is used in relatively unpacked data structures, is:

- Data type: Boolean
- Size: 1 octet

An all-zero value of the underlying Unsigned8 representation codes the value FALSE; any non-zero value codes the value TRUE.

12.22.3.2.3 Boolean1

The coding of a Boolean1, which is used in packed data structures, is:

- Data type: Boolean
- Size: 1 bit

A zero value of the underlying Unsigned1 representation codes the value FALSE, whereas the non-zero value one (1) codes the value TRUE.

12.22.3.3 Integer values

12.22.3.3.1 Coding of signed integer values

12.22.3.3.1.1 General

Signed integers are coded as 2's-complement numbers. In 2's-complement arithmetic, negative numbers are represented by the 2's-complement of the absolute value. In this system, zero has a single representation.

In the 2's-complement representation, positive numbers are represented as simple binary, and negative 2's-complement numbers are represented as the binary number that when added to a positive number of the same magnitude equals zero.

The most significant bit (i.e., bit 7 for an Integer8 value, bit 15 for an Integer16) indicates the sign of the integer, and is therefore called the sign bit. If the sign bit is zero, then the number represented is greater than or equal to zero (i.e., zero, or a positive number). If the sign bit is one, then the number represented is less than zero (i.e., a negative number).

NOTE To calculate the 2's-complement of an integer, the binary equivalent of the number is inverted by changing all of the ones to zeroes and all of the zeroes to ones (also called 1's-complement), and then one is added.

Example: The 2's-complement of the value 17 is formed.

0x 0001 000 1 (binary 17)

To form the 2's-complement:

First: NOT (0x 0001 000 1) = 0x 1110 111 0, where the NOT operation results in inverting the bits.

Then 1: (0x 1110 111 0) + (0x 0000 0001) = 0x 1110 1111 (2's-complement = -17) is added.

12.22.3.3.1.2 Integer8

The coding of an Integer8 is:

- Data type: Integer8
- Range: $-2^7 \leq k \leq 2^7 - 1$ (i.e., $-128 \leq k \leq 127$)
- Size: 1 octet

12.22.3.3.1.3 Integer16

The coding of an Integer16 is:

- Data type: Integer16
- Range: $-2^{15} \leq k \leq 2^{15} - 1$ (i.e., $-32\,768 \leq k \leq 32\,767$)
- Size: 2 octets

12.22.3.3.1.4 Integer32

The coding of an Integer32 is:

- Data type: Integer32
- Range: $-2^{31} \leq k \leq 2^{31}-1$ (i.e., -2 147 483 648< k < 2 147 483 647)
- Size: 4 octets

12.22.3.3.1.5 IntegerN

The coding of an IntegerN, which is used in packed data structures is:

- Data type: IntegerN
- Range: $-2^{-(N-1)} \leq k \leq 2^{(N-1)}-1$
- Size: N bits

12.22.3.3.2 Coding of unsigned integer values

12.22.3.3.2.1 Unsigned8

The coding of an Unsigned8 is:

- Data type: Unsigned8
- Range: $0 \leq k \leq 2^8-1$ (i.e., $0 \leq k \leq 255$)
- Size: 1 octet

Table 352 provides coding rules for Unsigned8 data.

Table 352 – Coding rules for Unsigned8

Octet	Bits							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

12.22.3.3.2.2 Unsigned16

The coding of an Unsigned16 is:

- Data type: Unsigned16
- Range: $0 \leq k \leq 2^{16}-1$ (i.e., $0 \leq k \leq 65\,535$)
- Size: 2 octets

Table 353 provides coding rules for Unsigned16 data.

Table 353 – Coding rules for Unsigned16

Octet	Bits							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8
2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

12.22.3.3.2.3 Unsigned32

The coding of an Unsigned32 is:

- Data type: Unsigned32
- Range: $0 \leq k \leq 2^{32}-1$ (i.e., $0 \leq k \leq 4\,294\,967\,295$)

- Size: 4 octets

Table 354 provides coding rules for Unsigned32 data.

Table 354 – Coding rules for Unsigned32

Octet	Bits							
	7	6	5	4	3	2	1	0
1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

12.22.3.3.2.4 Unsigned64

The coding of an Unsigned64 is:

- Data type: Unsigned64
- Size: 8 octets
- Range: $0 \leq k \leq 2^{64}-1$ (i.e., $0 \leq k \leq 18\ 446\ 744\ 073\ 709\ 551\ 615$)

Table 355 provides coding rules for Unsigned64 data.

Table 355 – Coding rules for Unsigned64

Octet	Bits							
	7	6	5	4	3	2	1	0
1	2^{63}	2^{62}	2^{61}	2^{60}	2^{59}	2^{58}	2^{57}	2^{56}
2	2^{55}	2^{54}	2^{53}	2^{52}	2^{51}	2^{50}	2^{49}	2^{48}
3	2^{47}	2^{46}	2^{45}	2^{44}	2^{43}	2^{42}	2^{41}	2^{40}
4	2^{39}	2^{38}	2^{37}	2^{36}	2^{35}	2^{34}	2^{33}	2^{32}
5	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
6	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

12.22.3.3.2.5 Unsigned128

The coding of an Unsigned128 is:

- Data type: Unsigned128
- Size: 16 octets
- Range: $0 \leq k \leq 2^{128}-1$ (i.e., $0 \leq k \leq$ 340 282 366 920 938 463 463 374 607 431 768 211 455)

Table 356 provides coding rules for Unsigned128 data.

Table 356 – Coding rules for Unsigned128

Octet	Bits							
	7	6	5	4	3	2	1	0
1	2^{127}	2^{126}	2^{125}	2^{124}	2^{123}	2^{122}	2^{121}	2^{120}
2	2^{119}	2^{118}	2^{117}	2^{116}	2^{115}	2^{114}	2^{113}	2^{112}
3	2^{111}	2^{110}	2^{109}	2^{108}	2^{107}	2^{106}	2^{105}	2^{104}
4	2^{103}	2^{102}	2^{101}	2^{100}	2^{99}	2^{98}	2^{97}	2^{96}
5	2^{95}	2^{94}	2^{93}	2^{92}	2^{91}	2^{90}	2^{89}	2^{88}
6	2^{87}	2^{86}	2^{85}	2^{84}	2^{83}	2^{82}	2^{81}	2^{80}
7	279	2^{78}	2^{77}	2^{76}	2^{75}	2^{74}	2^{73}	2^{72}
8	2^{71}	2^{70}	2^{69}	2^{68}	2^{67}	2^{66}	2^{65}	2^{64}
9	2^{63}	2^{62}	2^{61}	2^{60}	2^{59}	2^{58}	2^{57}	2^{56}
10	2^{55}	2^{54}	2^{53}	2^{52}	2^{51}	2^{50}	2^{49}	2^{48}
11	2^{47}	2^{46}	2^{45}	2^{44}	2^{43}	2^{42}	2^{41}	2^{40}
12	2^{39}	2^{38}	2^{37}	2^{36}	2^{35}	2^{34}	2^{33}	2^{32}
13	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
14	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
15	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
16	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

12.22.3.3.2.6 UnsignedN

The coding of an UnsignedN, which is used in packed data structures is:

- Data type: UnsignedN
- Range: $0 \leq k \leq 2^{N)}$-1
- Size: N bits

12.22.3.4 Floating point values

12.22.3.4.1 Coding of floating-point values

This standard uses the encoding defined by ISO/IEC/IEEE 60559 (based on IEEE 754) for normalized floating-point values and NaNs. Each value is represented by three contiguous fields:

- S, the sign of the floating-point value, where 0 and 1 represent positive and negative, respectively, conveyed in a 1-bit field;
- E, the exponent of the value, in base 2, plus a bias B, conveyed in a field occupying N_E = about 1/4 of the total number of bits of the representation of the floating-point value, where the value of B is $2^{(N_E-1)}$-1;
- F, the fractional part of the value's mantissa, also in base 2, conveyed in the remaining N_F bits of the value's representation.

When E is not all zero bits or all one bits, the resulting numeric value is $(-1)^S \times 2^{(E-B)} \times (1,F)$. When E and F are both all zero bits the value represented is a signed zero.

When E is all one bits and F is all zero bits the value represented is a signed infinity. See ISO/IEC/IEEE 60559 for further information regarding real number representation, range and precision, including encoding of signed zero, signed infinity (overflow), de-normalized numbers (underflow), and NaNs.

12.22.3.4.2 Single-precision float

Single-precision floating-point values are represented contiguously as shown in Table 357, where N_E = 8, B = 127 and N_F = 23. This permits a single-precision floating point value to be calculated by the following equation, which applies when E is not all zero bits or all one bits:

$$(-1)^S \times 2^{(E - 127)} \times (1,F)$$

Table 357 – Coding rules for single-precision float

Octet	Bits							
	7	6	5	4	3	2	1	0
	Sign (S)	Exponent (E)						
1	+/-	2^7	2^6	2^5	2^4	2^3	2^2	2^1
	(E)	Fraction (F)						
2	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
3	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}
4	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}	2^{-21}	2^{-22}	2^{-23}

12.22.3.5 Double-precision float

Double-precision floating-point values are represented contiguously as shown in Table 358, where N_E = 11, B = 1 023 and N_F = 52. This permits a double-precision floating point value to be calculated by the following equation, which applies when E is not all zero bits or all one bits:

$$(-1)^S \times 2^{(E - 1\,023)} \times (1,F)$$

Table 358 – Coding rules for double-precision float

Octet	Bits							
	7	6	5	4	3	2	1	0
1	Sign (S)	Exponent (E)						
	+/-	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4
2	Exponent (E) (continued)				Fraction (F)			
	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
3	Fraction (F) (continued)							
	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}
4	2^{-13}	2^{-14}	2^{-15}	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}
5	2^{-21}	2^{-22}	2^{-23}	2^{-24}	2^{-25}	2^{-26}	2^{-27}	2^{-28}
6	2^{-29}	2^{-30}	2^{-31}	2^{-32}	2^{-33}	2^{-34}	2^{-35}	2^{-36}
7	2^{-37}	2^{-38}	2^{-39}	2^{-40}	2^{-41}	2^{-42}	2^{-43}	2^{-44}
8	2^{-45}	2^{-46}	2^{-47}	2^{-48}	2^{-49}	2^{-50}	2^{-51}	2^{-52}

12.22.3.6 VisibleString

The coding of a visible string is:

- Type: VisibleString
- Range: See ISO/IEC 646 and ISO/IEC 2375: Defining registration number 2 + SPACE
- Coding: See ISO/IEC 646

NOTE See ISO/IEC 2375 for further details.

Table 359 provides coding rules for VisibleString data. If the size of the string is not determinable from other factors, then the size in octets is coded in one octet that immediately precedes the OctetString, as specified in Table 351.

Table 359 – Coding rules for VisibleString

Octet	Bits							
	7	6	5	4	3	2	1	0
1	First character in string							
2	Second character in string							
…	….							
N	Last character in string							

12.22.3.7 OctetString

The coding of an octet string is:

- Type: OctetString
- Coding: Binary

Table 360 provides coding rules for OctetString data. If the size of the string is not determinable from other factors, then the size in octets is coded in one octet that immediately precedes the OctetString, as specified in Table 351.

Table 360 – Coding rules for OctetString

Octet	Bits							
	7	6	5	4	3	2	1	0
1	First octet in string							
…	…							
N	Last octet in string							

12.22.3.8 BitString

The coding of a bit string that is not part of a superior packed structure is:

- Type: BitString
- Coding: Binary
- Size: Only multiples of 8 bits (i.e., multiples of octets) are supported for BitStrings that are not part of superior packed structures

Table 361 provides the general coding rule for BitString data. If the size of the string is not determinable from other factors, then the size in octets is coded in one octet that immediately precedes the BitString, as specified in Table 351.

Table 361 – Coding rules for BitString

Octet	Bits							
	7	6	5	4	3	2	1	0
1	(8 x N-1)$_{th}$	(8 x N-2)$_{th}$	(8 x N-3)$_{th}$	(8 x N-4)$_{th}$	(8 x N-5)$_{th}$	(8 x N-6)$_{th}$	(8 x N-7)$_{th}$	(8 x N-8)$_{th}$
	(bit position in string)							
2	(8 x N-9)$_{th}$	(8 x N-10)$_{th}$	(8 x N-11)$_{th}$	(8 x N-12)$_{th}$	(8 x N-13)$_{th}$	(8 x N-14)$_{th}$	(8 x N-15)$_{th}$	(8 x N-16)$_{th}$
…								
N	etc.							

12.22.3.9 SymmetricKey

A SymmetricKey is opaque. In this edition of this standard it is 128 bits. As such it is mapped, without interpretation, to an Octet16, which is sixteen octets in size.

12.22.4 Time-related data types

12.22.4.1 General

Time is continuous, potentially represented to nearly infinite precision in a nearly infinite range. Thus any reasonable representation of time has a specified finite resolution (e.g., 1 h, 1 s, 1 ns, 10^{-20} s, etc.) and a specified range, such as [0..1 d) or (0..10 000 yr], modulo which any value of time shall be represented.

Within this standard, TAINetworkTime is represented with a resolution of 2^{-16} s and a range of [0..2^{32}) s, modulo 2^{32} s. TAITimeRounded has the same range but rounds to the nearest 1 s and has a resolution of 2^{0} s (i.e., 1 s).

TAITimeDifference is intended for use to represent the difference between two different values of TAINetworkTime. That difference is also represented modulo 2^{32} s, so that very large numeric values likely represent negative differences. The determination of what part of the 2^{32} s range of a TAITimeDifference value is interpreted as a positive difference, versus the part that is interpreted as a negative difference, is determined by the use of that difference.

EXAMPLE When differencing two TAINetworkTime values during processing of a TPDU nonce, the specified logic specifically addresses differences in a small signed range and then classifies all other differences as “out of range” without attempting to assign them to either the relatively distant past or the relatively distant future relative to the referenced TAI time instant.

12.22.4.2 TAINetworkTime

TAINetworkTime represents the network time in TAI time as a six-octet fixed-point binary value with a resolution of 2^{-16} s modulo 2^{32} s. Thus the high-order four octets represent the current TAI time in units of 1 s while the low-order two octets represent the fractional TAI time in units of 2^{-16} s.

- Data type: TAINetworkTime

NOTE 1 This representation also applies to TAITimeDifference, which is a modulo difference.

- Valid range, expressed as an unsigned binary fixed-point value
  - whose integral component has the range 0..2^{32}-1 s (modulo 2^{32} s);
  - and whose fractional component has a resolution of 2^{-16} s.

NOTE 2 Because all possible values occur repeatedly (cyclically) in a modulo representation such as TAINetworkTime, it is not possible to code special-meaning values within this range, as can be done with the endpoints of a linear range.

Table 362 shows the representation for TAINetworkTime, and for TAITimeDifference when interpreted as a modulo difference.

Table 362 – Coding rules for TAINetworkTime, and for TAITimeDifference when interpreted as a modulo difference

Octet	Bits								Interpretation
	7	6	5	4	3	2	1	0	
1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Integral part of TAI time with granularity of 1 s
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
5	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	Fractional part of TAI time with granularity of 2^{-16} s
6	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}	2^{-16}	

12.22.4.3 TAITimeDifference

The coding of a TAITimeDifference is identical to that of TAINetworkTime. However, since it is a modulo value, it has potential interpretations as signed values. Those interpretations are:

- Data type: TAITimeDifference
- Valid range, expressed as a 2's-complement binary fixed-point value
  - whose integral component has the range $-2^{32}..2^{32}-1$ s;
  - whose fractional component has a resolution of 2^{-16} s; and
  - which is considered to "wrap" from positive to negative values at some Unsigned32 value for the integral component that is dependent on the specific usage scenario.

12.22.4.4 TAITimeRounded

TAITimeRounded represents the TAI time in integral seconds modulo the period of the representation, rounded to the nearest second. Its coding is:

- Data type: TAITimeRounded
- Valid Range: $0..2^{32}-1$ s (modulo 2^{32} s)

Table 363 shows the representation for TAITimeRounded.

Table 363 – Coding rules for TAITimeRounded

Octet	Bits								Interpretation
	7	6	5	4	3	2	1	0	
1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	TAI time with granularity of 1 s
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	

12.22.4.5 Standard data structures

Standard data structures are coded by concatenating the coded values for the structure elements in order from the lowest numbered element to the highest numbered element, beginning at octet 1 of the coded result.

12.22.4.6 Null

The data type null has a size of zero (0) octets. The value null is often used for semantic consistency, where it represents the potential for content when no content has been identified.

12.22.4.7 Packed

The data type packed indicates that one or more elements of the standard data types have been concatenated together without gap to maintain octet alignment. Additionally, packed elements of BitString and BooleanArray type need not occupy an integral number of octets. The structure and composition of packed data is implicitly known by the correspondents.

NOTE BooleanArrays are usually represented as packed BitStrings.

12.22.4.8 Structured data

12.22.4.8.1 SEQUENCE

SEQUENCE is used to indicate structured data of the same or different standard data type(s). This is akin to a record construct.

This standard does not support sequences that contain optionally-present members. If such a need is identified, a separate sequence (structure) shall be defined for each such required sequence of members. Correspondents are required to have prior knowledge of the structure of the sequence; thus no mechanism is provided to convey its structure explicitly.

12.22.4.8.2 SEQUENCE OF

For data, SEQUENCE OF is used to indicate an array construct. Array content may either be conveyed in its entirety, or a specified individual element of an array may be conveyed.

For conveyance of an individual element, the data type of the element is implicitly known by the correspondents. Since some data types are variable in size, the size of the element is conveyed with the element data.

When arrays are conveyed in their entirety, they are encoded in row-major-order. The size of the array in octets shall also be included. The data type of the elements is also known implicitly by the corresponding endpoints, and is not explicitly indicated in the APDU. The dimension(s) of the array is(are) also implicitly known by the corresponding endpoints, and hence is(are) not explicitly included in the APDU.

NOTE Following standard matrix notation, rows are identified by the first index of a two-dimensional array and columns by the second index. For example, for the “C” programming language, a two-dimensional array consisting of two rows and three columns, which visually would be

```
1  2  3
4  5  6 '
```

might be defined as

```
int A[2][3] = { {1, 2, 3}, {4, 5, 6} }
```

The encoding of this standard would convey the elements of this array in the following order: 1, 2, 3, 4, 5, 6.

12.22.4.8.3 CHOICE

CHOICE represents a selection chosen from a predefined enumeration of acceptable possibilities. Content of the data varies based on the choice selected.

12.22.4.8.4 OPTIONAL

OPTIONAL specifies that the designated component need not be included in the containing structure.

12.22.4.8.5 IMPLICIT

IMPLICIT specifies that those coding aspects that identify type, size, choice selection, and presence or absence as an optional element are to be suppressed when that information is otherwise determinable from context, such as from other elements of the data structure.

NOTE When a data type declaration ends with an explicit integer specifying the size of the atomic type (e.g., Unsigned12) or number of elements of an array type (e.g., OctetString2), that integer is implicit in the declaration and is not carried in the PDU as a size explicitly-conveyed within the item itself. Thus an OctetStringN does not contain a field specifying N, but an OctetString does contain such a field (because the size is not implicit in the declaration).

12.23 Syntax

12.23.1 Application protocol data unit

12.23.1.1 Start of containing module

NOTE 1 The object identifier root for the following definitions was changed to an IEC-based root to support correction and evolution of the TSDU structure relative to that of the original ISA 100.11a TSDU structure.

NOTE 2 The ASN.1 extensibility declaration "..." is used in each production that may be extended in future editions of this standard, or in industry-specific or vendor-specific ways for this edition.

```
IEC62734 (1 0 62734) edition (1) TSDU (1) ) DEFINITIONS
      IMPLICIT TAGS
      EXPORTS IEC62734_TSDU;
      ::= BEGIN
```

NOTE 3 For this edition of IEC 62734, the bit-level structure of IEC62734_TSDU is identical to that of ISA100_TSDU in ISA 100.11a:2011, 11.23, so either prefix designates a data structure with identical concrete representation and similar associated semantics. The equivalent prefix declaration for ISA 100.11a was

```
--   ( ISA ( ) ISA100.11a:2011 (71) ) DEFINITIONS
--     IMPLICIT TAGS
--     EXPORTS ISA100_TSDU;
--     ::= BEGIN
```

12.23.1.2 Top level definitions

```
IEC62734_TSDU :: = IMPLICIT CHOICE (
      individualAPDU                ASLIndividualAPDU,
      concatenatedAPDU              ASLConcatenatedAPDU
)
ASLIndividualAPDU ::= IMPLICIT CHOICE(
   confirmedRequestAPDU             ASLConfirmedServiceRequest,
   confirmedResponseAPDU            ASLConfirmedServicerResponse,
   unconfirmedRequestAPDU           ASLUnconfirmedServiceRequest,
   publicationAPDU                  ASLPublicationRequest
)
ASLConcatenatedAPDU ::= IMPLICIT SEQUENCE (
   IMPLICIT CHOICE (
      -- implicit based on the content of the APDU header, which is common across the choices
      confirmedRequest              ASLConfirmedServiceRequest,
      confirmedResponse             ASLConfirmedServiceResponse,
      unconfirmedRequest            ASLUnconfirmedServiceRequest
   )
)
```

NOTE This concatenation works because the size of each aperiodic APDU is either determined by explicit information or is implicit by service primitive definition.

12.23.1.3 Common substitutions

```
Float32 ::= REAL (WITH COMPONENTS(, base(2)) SIZE 32)          -- single-precision binary float
Float64 ::= REAL (WITH COMPONENTS(, base(2)) SIZE 64)          -- double-precision binary float

Integer8 ::= INTEGER (-128..127)                               -- 8-bit integer
Integer16 ::= INTEGER(-32 768..32 767)                         -- 16-bit integer
Integer32 ::= INTEGER(-4 294 967 296..4 294 967 295)           -- 32-bit integer

Unsigned8 ::= INTEGER(0..255)                                  -- 8-bit unsigned
Unsigned16 ::= INTEGER(0..65 535)                              -- 16-bit unsigned
Unsigned32 ::= INTEGER(0..4 294 967 295)                       -- 32-bit unsigned
Unsigned64 ::= INTEGER(0..18 446 744 073 709 551 615)          -- 64-bit unsigned
Unsigned128 ::= INTEGER(0..340 282 366 920 938 463 374 607 431 768 212 455)  -- 128-bit unsigned

Octet1 ::= Unsigned8
DL16Address ::= Unsigned16
EUI64Address ::= Unsigned64
IPv6Address ::= Unsigned128
SymmetricKey ::= PACKED ARRAY [128] OF BIT  -- opaque, uninterpretable bit string

TAINetworkTime::= SEQUENCE (                   -- referenced to TAI start time instant
   Seconds                                     Unsigned32,
   FractionalSecond                            Unsigned16
)

TAITimeRounded ::= SEQUENCE (                  -- referenced to TAI start time instant
   Seconds                                     Unsigned32,
)

TAITimeDifference ::= SEQUENCE (               -- not referenced to TAI start time instant
   Seconds                                     Unsigned32,
   FractionalSecond                            Unsigned16
)                                              -- See NOTE 1
```

NOTE 1 Since the representation of TAI time in this standard is modulo 2^{32} s, a value of type TAITimeDifference can be interpreted as either a positive or negative difference, with the two differing by 2^{32} s. Different uses of this type impose differing local limits on the expected range of numeric difference, which in turn determine how the modulo difference is interpreted. (E.g., -2^{31} s .. + (2^{31}-1) s, or -2^k s .. + (2^{32}-2^k-1) s for $0 \leq k < 32$, etc.)

NOTE 2 The following are only used within packed data structures.

```
Unsigned1 ::= INTEGER(0..1)                                    -- 1-bit unsigned
Unsigned2 ::= INTEGER(0..3)                                    -- 2-bit unsigned
Unsigned3 ::= INTEGER(0..7)                                    -- 3-bit unsigned
Unsigned4 ::= INTEGER(0..15)                                   -- 4-bit unsigned
Unsigned5 ::= INTEGER(0..31)                                   -- 5-bit unsigned
Unsigned6::= INTEGER(0..63)                                    -- 6-bit unsigned
Unsigned7 ::= INTEGER(0..127)                                  -- 7-bit unsigned

Unsigned9 ::= INTEGER(0..511)                                  -- 9-bit unsigned
Unsigned10 := INTEGER(0..1 023)                                -- 10-bit unsigned
Unsigned11 ::= INTEGER(0..2 047)                               -- 11-bit unsigned
Unsigned12 ::= INTEGER(0..4 095)                               -- 12-bit unsigned
Unsigned13 ::= INTEGER(0..8 191)                               -- 13-bit unsigned
Unsigned14 ::= INTEGER(0..16 383)                              -- 14-bit unsigned
Unsigned15 ::= INTEGER(0..32 767)                              -- 15-bit unsigned

Unsigned63 ::= INTEGER(0..9 223 372 036 854 775 807)           -- 63-bit unsigned
```

12.23.1.4 Application sublayer header

```
RequestResponse ::= Unsigned1 (
   request        (0),
   response       (1)
)

ObjectAddressingMode ::= Unsigned2 (
   compact        (0)          -- indicates 4-bit object identifiers
   midSize        (1)          -- indicates 8-bit object identifiers
   fullSize       (2)          -- indicates 16-bit object identifiers
   inferred       (3)          -- shall only be used as specified in 12.22.2.4.5.
)
```

```
ASLService ::= Unsigned5 (
   Publish                    0,
   AlertReport                1,
   AlertAcknowledge           2,
   Read                       3,
   Write                      4,
   Execute                    5,
   Tunnel                     6,
                                 -- values 7..31 are reserved for future use by this standard
)

ASLConfirmedServiceRequest ::= CHOICE(
   -- the first octet of the ConfirmedServiceRequest is constructed with
         -- bit 7 containing RequestResponse
         -- bits 6 and 5 containing ObjectAddressingMode
         -- bits 4..0 containing ASLService
   readCompact                [3]   IMPLICIT ReadRequestPDU,      -- bit pattern: 0000 0011
   readMidSize                [35]  IMPLICIT ReadRequestPDU,      -- bit pattern: 0010 0011
   readFull                   [67]  IMPLICIT ReadRequestPDU,      -- bit pattern: 0100 0011
   readInferred               [99]  IMPLICIT ReadRequestPDU,      -- bit pattern: 0110 0011
   writeCompact               [4]   IMPLICIT WriteRequestPDU,     -- bit pattern: 0000 0100
   writeMidSize               [36]  IMPLICIT WriteRequestPDU,     -- bit pattern: 0010 0100
   writeFull                  [68]  IMPLICIT WriteRequestPDU,     -- bit pattern: 0100 0100
   writeInferred              [100] IMPLICIT WriteRequestPDU,     -- bit pattern: 0110 0100
   executeCompact             [5]   IMPLICIT ExecuteRequestPDU,   -- bit pattern: 0000 0101
   executeMidSize             [37]  IMPLICIT ExecuteRequestPDU,   -- bit pattern: 0010 0101
   executeFull                [69]  IMPLICIT ExecuteRequestPDU,   -- bit pattern: 0100 0101
   executeInferred            [101] IMPLICIT ExecuteRequestPDU,   -- bit pattern: 0110 0101
   tunnelCompact              [6]   IMPLICIT TunnelRequestPDU,    -- bit pattern: 0000 0110
   tunnelMidSize              [38]  IMPLICIT TunnelRequestPDU,    -- bit pattern: 0010 0110
   funnelFull                 [70]  IMPLICIT TunnelRequestPDU,    -- bit pattern: 0100 0110
   tunnelInferred             [102] IMPLICIT TunnelRequestPDU     -- bit pattern: 0110 0110
)

ASLConfirmedServiceResponse ::= CHOICE(
   -- the first octet of the ConfirmedServiceResponse is constructed with
         -- bit 7 (MSBO containing RequestResponse) = 1 -- only response form is valid
         -- bits 6 and 5 containing ObjectAddressingMode
         -- bits 4..0 containing ASLService
   readCompact                [131]   IMPLICIT ReadResponsePDU,     --bit pattern: 1000 0101
   readMidSize                [163]   IMPLICIT ReadResponsePDU,     --bit pattern: 1010 0011
   readFull                   [195]   IMPLICIT ReadResponsePDU,     --bit pattern: 1100 0011
   readInferred               [227]   IMPLICIT ReadResponsePDU,     --bit pattern: 1110 0011
   writeCompact               [132]   IMPLICIT WriteResponsePDU,    --bit pattern: 1000 0100
   writeMidSize               [164]   IMPLICIT WriteResponsePDU,    --bit pattern: 1010 0100
   writeFull                  [196]   IMPLICIT WriteResponsePDU,    --bit pattern: 1100 0100
   writeInferred              [228]   IMPLICIT WriteResponsePDU,     --bit pattern: 1110 0100
   executeCompact             [133]   IMPLICIT ExecuteResponsePDU, --bit pattern: 1000 0101
   executeMidSize             [165]   IMPLICIT ExecuteResponsePDU, --bit pattern: 1010 0101
   executeFull                [197]   IMPLICIT ExecuteResponsePDU, --bit pattern: 1100 0101
   executeInferred            [229]   IMPLICIT ExecuteResponsePDU,  --bit pattern: 1110 0101
   tunnelCompact              [134]   IMPLICIT TunnelResponsePDU,   --bit pattern: 1000 0110
   tunnelMidSize              [166]   IMPLICIT TunnelResponsePDU,   --bit pattern: 1010 0110
   funnelFull                 [198]   IMPLICIT TunnelResponsePDU,   --bit pattern: 1100 0110
   tunnelInferred             [230]   IMPLICIT TunnelResponsePDU   --bit pattern: 1110 0110
)

ASLUnconfirmedServiceRequest ::= CHOICE (
   -- the first octet of the UnconfirmedServiceRequest is constructed with
         -- bit 7 (MSBO containing RequestResponse) = 0 -- only request form is valid
         -- bits 6 and 5 containing ObjectAddressingMode
         -- bits 4..0 containing ASLService
   alertReportCompact         [1]   IMPLICIT AlertReportRequestPDU,    --bit pattern: 0000 0001
   alertReportMidSize         [33]  IMPLICIT AlertReportRequestPDU,    --bit pattern; 0010 0001
   alertReportFull            [65]  IMPLICIT AlertReportRequestPDU,    --bit pattern: 0100 0001
   alertReportInferred        [97]  IMPLICIT AlertReportRequestPDU,     --bit pattern: 0110 0001
   alertAcknowledgeCompact    [2]   IMPLICIT AlertAcknowledgeRequestPDU, --0x 0000 0010
   alertcknowledgeMidSize     [34]  IMPLICIT AlertAcknowledgeRequestPDU, --0x 0010 0010
   alertReportFull            [66]  IMPLICIT AlertAcknowledgeRequestPDU, --0x 0100 0010
   alertReportInferred        [98]  IMPLICIT AlertAcknowledgeRequestPDU, --0x 0110 0010
   tunnelCompact              [6]   IMPLICIT TunnelRequestPDU,     --bit pattern: 0000 0110
   tunnelMidSize              [38]  IMPLICIT TunnelRequestPDU,     --bit pattern: 0010 0110
   funnelFull                 [70]  IMPLICIT TunnelRequestPDU,     --bit pattern: 0100 0110
   tunnelInferred             [102] IMPLICIT TunnelRequestPDU     --bit pattern: 0110 0110
)
```

```
ASLPublicationServiceRequest ::= CHOICE (
   -- the first octet of the PublicationServiceRequest is constructed with
         -- bit 7 (MSBO containing RequestResponse) = 0 -- only request form is valid for publication)
         -- bits 6 and 5 containing ObjectAddressingMode
   publishCompact                [0]   IMPLICIT PublishRequestPDU,   bit pattern: 0000 0000
   publishMidSize                [32]  IMPLICIT PublishRequestPDU,   bit pattern: 0010 0000
   publishFull                   [64]  IMPLICIT PublishRequestPDU    bit pattern: 0100 0000
                  -- inferred addressing is not used as there is no concatenation of publications
                  --  (see concentrator / dispersion objects)
)
```

12.23.1.5 Individual APDUs

```
SourceAndDestinationOIDs:: = IMPLICIT SEQUENCE (OCTET ALIGNED)(
      IMPLICIT CHOICE ( -- as determined by objectAddressingMode in bits 5 and 6 of first octet of APDU
            -- source object represents the initiator of the service primitive (.req or .rsp)
            -- destination object represents the recipient of the service primitive (.ind or .cnf)
            compact IMPLICIT PACKED SEQUENCE (
                  compactSourceObject        Unsigned4,
                  compactDestinationObject   Unsigned4
            )
            midSize IMPLICIT SEQUENCE (
                  midSizeSourceOID           Unsigned8,
                  midSizeDestinationOID      Unsigned8
            )
            fullSize IMPLICIT SEQUENCE (
                  fullSizeSourceOID          Unsigned16,
                  fullSizeDestinationOID     Unsigned16
            )
            inferred NULL
      )

ReadRequestPDU ::= IMPLICIT SEQUENCE
   soidDoid                         SourceAndDestinationOIDs,
   readRequest                      ReadRequest
)

ReadResponsePDU ::= IMPLICIT SEQUENCE
   soidDoid                         SourceAndDestinationOIDs,
   readResponse                     ReadResponse
)

WriteRequestPDU ::= IMPLICIT SEQUENCE
   soidDoid                         SourceAndDestinationOIDs,
   writeRequest                     WriteRequest
)

WriteResponsetPDU ::= IMPLICIT SEQUENCE
   soidDoid                         SourceAndDestinationOIDs,
   writeResponse                    WriteResponse
)

ExecuteRequestPDU ::= IMPLICIT SEQUENCE
   soidDoid                         SourceAndDestinationOIDs,
   executeRequest                   ExecuteRequest
)

ExecuteResponsePDU ::= IMPLICIT SEQUENCE
   soidDoid                         SourceAndDestinationOIDs,
   executeResponse                  ExecuteResponse
)

TunnelRequestPDU ::= IMPLICIT SEQUENCE
   soidDoid                         SourceAndDestinationOIDs,
   tunnelRequest                    TunnelRequest
)

TunnelResponsePDU ::= IMPLICIT SEQUENCE
   soidDoid                         SourceAndDestinationOIDs,
   tunnelResponse                   TunnelResponse
)
```

```
AlertReportRequestPDU ::= IMPLICIT SEQUENCE
   soidDoid                          SourceAndDestinationOIDs,
   alertReportRequest                AlertReportRequest
)

AlertAcknowledgeRequestPDU ::= IMPLICIT SEQUENCE
   soidDoid                          SourceAndDestinationOIDs,
   alertAcknowledgeRequest           AlertAcknowledgeRequest
)

PublishRequestPDU ::= IMPLICIT SEQUENCE
   soidDoid                          SourceAndDestinationOIDs,
   publishRequest                    PublishRequest
)
```

12.23.1.6 Periodic APDUs

```
PublishRequest ::= IMPLICIT SEQUENCE (
   IMPLICIT CHOICE ( -- implicitly determined by the corresponding application processes
      NativeValue                    IMPLICIT PublishedValue,              -- single published value
      NativeSequence                 IMPLICIT PublishedValueSequence,   -- sequence of published values
      HealthReportSequence           IMPLICIT HealthReportSequence,        -- publication HRCO
      nonNativeSequence              IMPLICIT NonNativeSequence            -- publication tunnel
       )
)

PublishedValue ::= IMPLICIT SEQUENCE (
   contentVersion                    Unsigned8,    -- version of configuration of content published
   freshValueSequenceNumber          Unsigned8,    -- freshness of this set of data
   value                             ProcessValueAndStatus
)

PublishedValueSequence ::= IMPLICIT SEQUENCE (
   contentVersion                    Unsigned8,    -- version of configuration of content published
   freshValueSequenceNumber          Unsigned8,    -- freshness of this set of data
   publishedValues                   SEQUENCE OF ProcessValueAndStatus
)

HealthReportSequence ::= IMPLICIT SEQUENCE (
   contentVersion                    Unsigned8,    -- version of configuration of content published
   freshValueSequenceNumber          Unsigned8,    -- freshness of this set of data
   healthReportSize                  Unsigned8,
   healthReport                      OCTET STRING
)

NonNativeSequence ::= IMPLICIT OCTET STRING

ProcessValueAndStatus ::= IMPLICIT CHOICE ( -- based on publisher and subscriber application configuration
   analog         AnalogProcessValueAndStatus,
   boolean        BooleanProcessValueAndStatus
                  -- NOTE   This choice element can be extended by industry consortia and vendors
)

AnalogProcessValueAndStatus ::= IMPLICIT SEQUENCE (
      valueStatus                    PV_Status,
      analogProcessValue             Float32
)

BooleanProcessValueAndStatus ::= IMPLICIT SEQUENCE (
      valueStatus                    PV_Status,
      booleanProcessValue            Boolean8      -- single Boolean value represented by a full octet
)
```

```
PV_Status ::= PACKED SEQUENCE (OCTET ALIGNED) ( -- 1 octet (bit field sizes are: 2 + 1 + 3 + 2)
   quality                                PV_Quality,              -- 2 bits
   reservedSpareBit                       Unsigned1,               -- 1 bit
   IMPLICIT CHOICE ( -- selected by quality; all are               -- 3 bits
      [0] BadValueSubstatus               BadValueSubstatus,
      [1] UncertainValueSubstatus         UncertainValueSubstatus,
      [2] GoodValueSubstatus              GoodValueSubstatus,
      [3] otherSubstatus                  Unsigned3                          -- reserved for future use
    ),                                                                       -- 1 spare code point
   limitStatus                            LimitStatus              -- 2 bits control anti-windup information
)

PV_Quality ::= Unsigned2 (                      -- 2 bits
   badValue,                           (0),                                  -- value is bad
   uncertainValue                      (1),                                  -- value is uncertain
   goodValue                           (2),                                  -- value is good
   otherValue                          (3)                                   -- reserved for future use
)                                                                            -- 1 spare code point

BadValueSubstatus ::= Unsigned3 (               -- 3 bits
   badValue_NonSpecific,               (0),
   badValue_ConfigurationError,        (1),
   badValue_NotConnected               (2),
   badValue_DeviceFailure,             (3),
   badValue_SensorFailure,             (4),
   badValue_NoCommunicationWithLUV     (5),
   badValue_NoCommunicationNoLUV       (6),
   badValue_OutOfService               (7)
)                                                                            -- no spare code points

UncertainValueSubstatus ::= Unsigned3 ( -- 3 bits
   uncertainValue_NonSpecific,         (0),
   uncertainValue_LastUsableValue      (1),
   uncertainValue_SubstitutedOrManualEntry (2),
   uncertainValue_InitialValue         (3),
   uncertainValue_SensorConversionInaccurate, (4),
   uncertainValue_RangeLimitsExceeded  (5)
   uncertainValue_SubNormal,           (6),
   uncertainValue_Spare                (7)                                   -- reserved for future use
)                                                                            -- 1 spare code point

GoodValueSubstatus ::= Unsigned3 (      -- 3 bits
   goodValue_NoSpecialConditionsExist  (0),
   goodValue_SpecialCondition1         (1),                                  -- reserved for future use
   goodValue_SpecialCondition2         (2),                                  -- reserved for future use
   goodValue_SpecialCondition3         (3),                                  -- reserved for future use
   goodValue_SpecialCondition4         (4),                                  -- reserved for future use
   goodValue_SpecialCondition5         (5),                                  -- reserved for future use
   goodValue_SpecialCondition6         (6),                                  -- reserved for future use
   goodValue_SpecialCondition7         (7)                                   -- reserved for future use
)                                                                            -- 7 spare code points

LimitStatus ::= Unsigned2 ( -- 2 bits
   notLimited                          (0),
   lowLimited                          (1),
   highLimited                         (2),
   constant                            (3)     -- both high limited and low limited
)                                                                            -- no spare code points

highLowLimited LimitStatus ::= LimitStatus     constant          -- alternative symbolic name
lowHighLimited LimitStatus ::= LimitStatus     constant          -- alternative symbolic name
```

12.23.1.7 Aperiodic APDUs

```
CompactObjectIdentifier ::= Unsigned4
MidSizeObjectIdentifier ::= Unsigned8
FullSizeObjectIdentifier ::= Uinsigned16
```

```
ExtensibleInteger ::= IMPLICIT SEQUENCE (OCTET ALIGNED) (
    format                          Boolean1, -- 1 bit, FALSE for short form, TRUE for long form
    IMPLICIT CHOICE ( -- choice is established by the format field
      shortForm                     Unsigned7,          -- 7 bits       -- value shall be < 0x80
      longForm                      Unsigned15,         -- 15 bits      -- value shall be ≥ 0x80 and
     )                                                                  -- < 0x800; value < 0x80 are invalid
)
```

An ExtensibleInteger shall use a minimal-size encoding. Use of a longForm to encode a value that could be encoded as a shortForm is invalid and shall be rejected as a protocol error.

```
AttributeClass ::= Unsigned2 ( -- code points for attribute alternatives
    sixBitNoIndexing                (0),    -- 6-bit attribute identifier, no index
    sixBitOneDimension              (1),    -- 6-bit attribute identifier, one index (8 or 16 bits)
    sixBitTwoDimensions             (2),    -- 6-bit attribute identifier, two indices (each 8 or 16 bits)
    twelveBitExtended               (3)     -- 12-bit attribute identifier
)

TwelveBitIndexClass ::= Unsigned2 ( – code points for 12-bit AID indexing alternatives
    twelveBitNoIndexing             (0),
    twelveBitOneDimension           (1),
    twelveBitTwoDimensions          (2),
    twelveBitReserved               (3)
)

ExtensibleAttributeIdentifier ::= IMPLICIT PACKED SEQUENCE (OCTET ALIGNED) (
    attributeFormat                 AttributeClass          --2 bits
    IMPLICIT CHOICE ( -- choice is established by element attributeFormat
        sixBitNoIndexing            Unsigned6,
        sixBitOneDimension IMPLICIT SEQUENCE (OCTET ALIGNED) (
            sixBitOneIndexAID       Unsigned6,
            sixBitOneIndex          ExtensibleInteger,
        ),
        sixBitTwoDimensions IMPLICIT SEQUENCE (OCTET ALIGNED) (
            sizBitTwoIndexAID       Unsigned6,
            sixBitTwoIndexNo1       ExtensibleInteger,
            sixBitTwoIndexNo2       ExtensibleInteger
        ),
        twelveBitExtended IMPLICIT SEQUENCE (OCTET ALIGNED) (
            twelveBitIndex          TwelveBitIndexClass,
            twelveBitAID            Unsigned12
            CHOICE ( -- choice is established by the twelveBitIndexClass
                                    twelveBitNoIndexing NULL,
                                    twelveBitOneDimension : ExtensibleInteger,
                                    twelveBitTwoDimensions IMPLICIT SEQUENCE (OCTET ALIGNED) (
                                        TwelveBitTwoIndexNo1        ExtensibleInteger,
                                        TwelveBitTwoIndexNo2        ExtensibleInteger
                                    )
            )
        )
    )
)
```

NOTE The four bits in the first octet and eight bits of the second octet of the attributeID are concatenated to form a longer Unsigned12 value when the 12-bit attributeID alternative is indicated. The four bits in the first octet are the most significant, and the eight bits in the second octet are the least significant.

```
ScalarType :::= Unsigned12 (
   Null                    (0)
   Boolean8                (1),    -- single Boolean value represented by a full octet
   Integer8                (2),
   Integer16               (3),
   Integer32               (4),
   Unsigned8               (5).
   Unsigned16              (6).
   Unsigned32              (7),
   Float32                 (8),
   VisibleString           (9),    -- GenericSizeAndValue format
   OctetString             (10),   -- GenericSizeAndValue format


   BitString               (14),

   Float64                 (30),
   TAITimeDifference       (31),
   TAINetworkTime          (32)
)                                  -- all other code points are reserved for this standard
```

Primitive encoding shall be used for ScalarData, ArrayData, and StructureData value elements. No type information is included in the encoding.

```
GenericSizeAndValue ::= IMPLICIT SEQUENCE OF (
      SizeInOctets             ExtensibleInteger,      -- necessary for parsing (e.g., concatenations)
      DataValue                IMPLICIT SEQUENCE OF Octet1
 )

ServiceFeedbackCodeGenericSizeAndValue ::= GenericSizeAndValue
```

12.23.2 Alert reports and acknowledgments

```
AlertClass ::= Unsigned1 (                                  -- 1 bit
   event                   (0),
   alarm                   (1)
 )

AlertCategory ::= Unsigned2 ( -- 2 bits
   deviceDiagnostic        (0),
   communicationsDiagnostic (1),
   security                (2),
   process                 (3)
 )

AlarmDirection ::= Unsigned1 ( -- 1 bit
   returnToNormalOrNoAlarm (0),    -- for alerts, set this value to 0; for alarm returns set this to zero
   inAlarm                 (1)     -- to report an alarm condition, set this value to 1.
)
```

This standard presently does not define standard alerts for the following industry-independent AL-defined objects:

– UAPMO;
– ARO;
– UDO;
– Concentrator;
– Dispersion;
– Tunnel;
– Interface.

```
ASLMO_Communication_Alerts ::= ENUMERATED (
   malformed_APDU                    (0),
                  -- values 1..50 are reserved for future use by this standard
                  -- values 51..100 are reserved for future use by standard profiles
                  -- vendor-specific codes range 101..255
)

AI_ProcessAlerts ::= ENUMERATED ( -- 1 octet;
   outOfServiceAlarm                 (0),
   highAlarm                         (1),
   highHighAlarm                     (2),
   lowAlarm                          (3),
   lowLowAlarm                       (4),
   deviationLowAlarm                 (5),
   deviationHighAlarm                (6)
                  -- values 7..50 are reserved for future use by this standard
                  -- values 51..100 are reserved for future use by standard profiles
                  -- vendor-specific codes range 101..255
)

AO_ProcessAlerts ::= ENUMERATED ( -- 1 octet;
   outOfServiceAlarm                 (0),
   highAlarm                         (1),
   highHighAlarm                     (2),
   lowAlarm                          (3),
   lowLowAlarm                       (4),
   deviationLowAlarm                 (5),
   deviationHighAlarm                (6)
                  -- values 7..50 are reserved for future use by this standard
                  -- values 51..100 are reserved for future use by standard profiles
                  -- vendor-specific codes range 101..255
 )

BI_ProcessAlerts ::= ENUMERATED ( -- 1 octet;
   outOfServiceAlarm                 (0),
   discreteAlarm                     (1)
                  -- values 2..50 are reserved for future use by this standard
                  -- values 51..100 are reserved for future use by standard profiles
                  -- vendor-specific codes range 101..255
 )

BO_ProcessAlerts ::= ENUMERATED (
   outOfServiceAlarm                 (0),
   discreteAlarm                     (1)
                  -- values 2..50 are reserved for future use by this standard
                  -- values 51..100 are reserved for future use by standard profiles
                  -- vendor-specific codes range 101..255
 )

ARMO_Alerts ::= ENUMERATED (
   AlarmRecoveryStart                (0),
   AlarmRecoveryEnd                  (1)
                  -- values 2..50 are reserved for future use by this standard
                  -- values 51..100 are reserved for future use by standard profiles
                  -- vendor-specific codes range 101..255
)

IndividualAlertID :: = Unsigned8                  -- unique ID associated with an individual alert
                                                  --  assigned by the application process in the UAL

statusSignalNamur107  ::= Unsigned8 (
   failure                           (0),    --
   checkFunction                     (1),    --
   offSpec                           (2),    --
   maintenanceRequired               (3),    --
)
```

```
IndividualAlert ::= IMPLICIT PACKED SEQUENCE (OCTET ALIGNED)(
      individualAlertID                 IndividualAlertID,
      DetectingObjectTransportLayerPort Unsigned16,
      DetectingObject                   Unsigned16,
      DetectingObjectType               Unsigned16,
      detectionTimeTAINetworkTime,      -- 48 bits
      alertClass                        AlertClass,          -- 1 bit
      alarmDirection                    AlarmDirection,      -- 0: event or alarm return; 1: alarm report
      alertCategory                     AlertCategory,       -- 2 bits: device, comm, security, process
      alertPriority                     AlertPriority,       -- 4 bits
      alertType                         Unsigned8,           -- object category and type dependent
      associatedDataSize                ExtensibleInteger,
      associatedData                     -- present when associatedDataSize > 0
         CHOICE  ( -- choice is based on AlertCategory
            communicationsDiagnostic IMPLICIT SEQUENCE OF Octet1 OPTIONAL,
            security  IMPLICIT SEQUENCE OF Octet1 OPTIONAL,
            process IMPLICIT SEQUENCE OF Octet1 OPTIONAL,
            deviceDiagnostic IMPLICIT SEQUENCE
                  (  namur107Status          statusSignalNamur107,
                     detailedInformation     IMPLICIT SEQUENCE OF Octet1 OPTIONAL
                  )                              -- may include additional information per NAMUR-107
            ) OPTIONAL
)

AlertReportRequest ::= ( -- note: client OID not present; ARMO is implied
      alert          IndividualAlert
 )

AlertAcknowledgeRequest ::= (
      alertID        IndividualAlertID                  -- server is always ARMO
 )

AlertPriority ::= Unsigned4
```

Alert priority is a value that indicates the importance of the alert. A larger value implies a more important alert. Host systems map device priorities into host alert priorities that usually include the categories:

– urgent,

– high,

– medium,

– low, and

– journal.

The recommended mapping of alert priority values into these categories is specified in 12.17.5.2.2.22.

```
MalformedAPDUClass ::= AlertClassevent;

MalformedAPDUAlertCategory ::= AlertCategorycommunicationsDiagnostic

MalformedAPDUAlertType ::= AlertTypemalformedAPDUCommunicationAlert

MalformedAPDUAlertPriority = 7              -- mid-range of medium priority alerts

MalformedPDUAlertValue ::= IMPLICIT SEQUENCE ( -- alert value sent by ASL to DMAP
   sourceAddress                    IPv6Address,          -- 128 bits to ensure address uniqueness.
   thresholdExceeded                Unsigned16,
   TimeWindow                       TAITimeDifference
 )

MalformedPDUAlertValueSize ::= 24       -- sizeof(MalformedPDUAlertValue)
```

12.23.3 Service feedback code

NOTE Service feedback code is used to indicate status (e.g., success), warning (e.g., value limited), or error (e.g., incompatible mode).

```
ServiceFeedbackCode ::= Unsigned8 (      -- 1octet
   -- standard error codes, range 0..127
   success                               (0)     -- success
   failure                               (1)     -- generic failure
   other                                 (2),    -- reason other than that listed in this enumeration
   invalidArgument                       (3),    -- invalid attribute to a service call
   invalidObjectID                       (4),    -- invalid object ID
   invalidService                        (5),    -- unsupported or illegal service
   invalidAttribute                      (6),    -- invalid attribute index
   invalidElementIndex                   (7),    -- invalid array or structure element index (or indices)
   readOnlyAtrribute                     (8),    -- read-only attribute
   valueOutOfRange                       (9),    -- value is out of permitted range
   inappropriateProcessMode              (10),   -- process is in an inappropriate mode for the request
   incompatibleMode                      (11),   -- value is not acceptable in current context

   invalidValue                          (12),   -- value (data) not acceptable for other reason
                                                 (e.g., too large, too small, invalid engineering units code)
   internalError                         (13),   -- device internal problem
   invalidSize      (14),                        -- size is not valid (may be too big or too small)
   incompatibleAttribute                 (15),   -- attribute not supported in this version
   invalidMethod                         (16),   -- invalid method identifier
   objectStateConflict                   (17),   -- state of object in conflict with action requested
   inconsistentContent                   (18),   -- the content of the service requested is inconsistent
   invalidParameter                      (19),   -- value conveyed is not legal for method invocation
   objectAccessDenied                    (20),   -- object is not permitting access
   typeMismatch                          (21),   -- data not as expected (e.g., too many or too few octets)
   deviceHardwareCondition               (22),   -- device specific hardware condition prevented request from
                                                    succeeding (e.g., memory defect problem)
   deviceSensorCondition                 (23),   -- problem with sensor detected
  deviceSoftwareCondition                (24),   -- device specific software condition prevented request from
                                                 succeeding (e.g., local lockout, local write protection,
                                                 --   simulating in progress)
   fieldOperationCondition               (25),   -- field specific condition prevented request from succeeding
                                                 (e.g., lockout, or environmental condition not in range)
   configurationMismatch                 (26),   -- a configuration conflict was detected
   insufficientDeviceResources           (27),   -- e.g., queue full, buffers/memory unavailable
   valueLimited                          (28),   -- e.g., value limited by device
   dataWarning                           (29),   -- e.g., value has been modified due to a device specific reason
   invalidFunctionReference              (30),   -- function referenced for execution is invalid
   functionProcessError                  (31),   -- function referenced could not be performed due to a device
                                                 specific reason

   warning                               (32),   -- successful, but there is additional information that may be of
                                                 interest to the user which may, for example be conveyed via
                                                 accessing an attribute or by sending an alert
  writeOnlyAttribute                     (33),   -- write-only attribute (e.g., a command attribute)
   operationAccepted                     (34),   -- method operation accepted
   invalidBlockSize                      (35),   -- upload or download block size not valid
   invalidDownloadSize                   (36),   -- total size for upload not valid
   unexpectedMethodSequence              (37),   -- required method sequencing has not been followed
   timingViolation                       (38),   -- object timing requirements have not been satisfied
   operationIncomplete                   (39),   -- method operation, or method operation sequence not
                                                 successful
   invalidData                           (40),   -- data received is not valid
                                                 --  (e.g., checksum error, data content not as expected, etc.)

   dataSequenceError                     (41),   -- data is ordered; data received is not in the order required
                                                    example: duplicate data was received
   operationAborted                      (42),   -- operation aborted by server
   invalidBlockNumber                    (43),   -- invalid block number
   blockDataError                        (44),   --error in block of data, example, wrong size, invalid content
   blockNotDownloaded                    (45),   -- the specified block of data has not been successfully
                                                 downloaded
   writeProtected                        (46),   -- data is write protected, so write operation is invalid
   invalidMode                           (47),   -- operation did not succeed due to invalid mode
      -- ...                                    -- range 48..127 is reserved for future use of this standard
                                                 -- vendor-specific device-specific feedback codes, range 128..255
   vendorDefinedError_128                (128),  -- redefinable by each device vendor for each device type
      -- ...
   vendorDefinedError_254                (254),  -- redefinable by each device vendor for each device type
   extensionCode                         (255)   -- indicates a two-octet field size for an extended service
                                                    feedback code value
)                                                -- 123 values redefinable by each device vendor for each device type
```

12.23.4 Read, write, and execute

```
RequestID ::= Unsigned8
ReadRequest ::= IMPLICIT SEQUENCE (
   requestID                          RequestID,
   targetAttribute                    ExtensibleAttributeIdentifier
)

ApduResponseControlData ::= PACKED IMPLICIT SEQUENCE (
   ,
   Spare                              Unsigned7,          -- redefinable in future editions of this standard
   ForwardCongestionNotificationEcho  Boolean1  -- TRUE when congestion in forward (request) path detected
)

ReadResponse ::= IMPLICIT SEQUENCE (
   requestID                          RequestID,          -- matches corresponding ReadRequest
   apduControl                        ApduResponseControlData,
   readValue                          ServiceFeedbackCodeGenericSizeAndValue
)

WriteRequest ::= IMPLICIT SEQUENCE (
   requestID                          RequestID,
   targetAttribute                    ExtensibleAttributeIdentifier
   value                              GenericSizeAndValue
)

WriteResponse ::= IMPLICIT SEQUENCE (
   requestID                          RequestID,          -- matches corresponding WriteRequest
   apduControl                        ApduResponseControlData,
   serviceFeedbackCode                ServiceFeedbackCode
)

MethodInvocationRequest ::= IMPLICIT SEQUENCE (
   methodID                           Unsigned8,
   requestParametersSize              ExtensibleInteger,
   requestParameters                  IMPLICIT SEQUENCE of Octet1 OPTIONAL
      -- primitive encoding; data type known by correspondents
      -- requestParameters only present if requestParametersSize >0
)

MethodInvocationResponse ::= IMPLICIT SEQUENCE (
   responseParametersSize             ExtensibleInteger,
   responseParameters                 IMPLICIT SEQUENCE of Octet1 OPTIONAL
      -- primitive encoding; data type known by correspondents
      -- responseParameters only present if responseParametersSize >0
)

ExecuteRequest :: = IMPLICIT SEQUENCE (
      requestID                       RequestID,
      methodinvocationRequest         MethodInvocationRequest    -- data type(s) specified by standard
)

ExecuteResponse ::= IMPLICIT SEQUENCE (
      requestID                       RequestID,
      apduControl                     ApduResponseControlData,
      serviceFeedbackCode             ServiceFeedbackCode,
      methodInvocationResponse        MethodInvocationResponse    -- data type(s) specified by standard
)
```

12.23.5 Tunnel

```
TunnelRequest ::= IMPLICIT SEQUENCE (
   length                             ExtensibleInteger,
   tunnelPayload                      SEQUENCE OF Octet1
)

TunnelResponse ::= IMPLICIT SEQUENCE (
   apduControl                        ApduResponseControlData,
   length                             ExtensibleInteger,
   tunnelPayload                      SEQUENCE OF Octet1
)
```

12.23.6 End of contained module

END

12.24 Detailed coding examples (informative)

12.24.1 Read

Scenario: Client object 11 wishes to read data from server object 12, attribute 3. The response indicates the read is successful and returns a value of size two octets.

Table 364 illustrates an example of a request to read multiple values.

Table 364 – Coding example: Read request for a non-indexed attribute

Encoding of octets in hexadecimal	Semantic
03	Read request
BC	Client (source) object ID = 11_{10} Server (destination) object ID = 12_{10}
XX	Request identifier
03	Attribute ID = 3 (attribute is scalar)

Table 365 illustrates an example of a response to a request to read multiple values.

Table 365 – Coding example: Read response for a non-indexed attribute

Encoding of octets in hexadecimal	Semantic
83	Read response
CB	Server (source) object iD = 12_{10} Client (destination) object ID = 11_{10}
XX	Request identifier (same value as for Request identifier that was included in the corresponding service request)
00	Success
02	Value is two octets long
YY YY	Value

12.24.2 Tunnel

Scenario: Object 16 in the client is sending a message to object 20 in the server. The content of the message is to be passed through to the server object.

Table 366 illustrates an example of a tunnel service request that has payload size of 9 octets.

Table 366 – Coding example: Tunnel service request

Encoding of octets in hexadecimal	Semantic
06	Tunnel request
09	Size
(9 octets of tunneled data)	Data being tunneled

13 Provisioning

13.1 General

A device conforming to this standard is considered provisioned when the device has the information required to communicate with a target network and initiate a join request to the system/security manager of the target network. In this document, a target network is defined as a network the device is being provisioned to join. The information required to initiate the join request includes both security (trust-related) information and network-related information. Clause 13 specifies the over-the-air provisioning procedure and message format where the Type A field medium is used and out-of-band message formats where the Type A field medium is not used to provision the trust-related and network-related information.

Over-the-air provisioning uses the subnet joining process to set up a connection between the provisioning device and the device being provisioned. The subnet joining process is described in 6.3.9.2 and follows two optional paths, one defined for a device joining with trust-related information based on a symmetric key, and another defined for a device joining with trust-related information based on an asymmetric key. Out-of-band provisioning may not use the subnet joining process; it can instead provision the information via another wired or wireless means.

The goal of the provisioning process is to provide enough information so that one of these paths can be taken by the device.

The provisioning process involves a device that implements the provisioning role by providing the network-related and trust-related information to the new device. During provisioning, the operator can use the provisioning device, acting as a proxy for the system manager, to decide if a new device should be connected to the network or not, with information from the security manager. In another example, a copy of the list of allowed devices can be obtained from the security manager, allowing the provisioning device to make a local decision. When the target network is a secure network both trust-related and network-related information needs to be provisioned; for unsecured networks the default key (K_global) is used as the trust-related information. Once a device is provisioned, it is ready to join the target network. Thereafter, usually without human intervention, the security manager of the target network may either accept or reject the join request to the target network from the device based on the provisioned information.

NOTE In this standard, various aspects related to installation of the trust-related and network-related information in a device, conveyance of this information to the security manager, and establishment of trust are described in different clauses. Installation of the trust-related and network-related information is described in Clause 13. Conveyance of the information to the security manager is described in Clause 9 and Clause 10. Establishment of trust is described in 7.4.4.3.2.

13.2 Terms and definitions for devices with various roles or states

The following terms are defined for devices with various roles or states:

- **Device being provisioned (DBP)**: A device that needs to be provisioned, or is in the process of being provisioned. The device may be missing all or part of the information required to join a network.

NOTE 1 A device that contains old information relating to a network often is updated by provisioning it with new information.

- **Target network**: The network that the DBP is being provisioned to join.
- **Provisioning device (PD)**: A device that implements the role of provisioning another device to allow that device to join the target network. A PD need not be a device implementing only the provisioning role; rather, it could be:
  - the system/security manager of the target network;

NOTE 2 The system/security manager role is distributable, e.g., to a designated set of devices in the target network.

  - a device, such as a handheld device containing a system/security manager, that uses the protocol suite specified by this standard to provision the DBP through a separate, temporary mini-network; or
  - a device that uses out-of-band (OOB) communication, such as infrared, near field communications (NFC), or plugs, to provision the DBP, where the OOB communication is outside the scope of this standard.

- **Default network**: The network whose network identifier is 1.
- **Provisioning network**: A network formed between the PD and the DBP. If the PD is a handheld, then the provisioning mini-network is the network formed between the handheld and the DBP. If the provisioning device is the security manager of the target network, then the provisioning network may be a separate logical network on the target network itself.
- **Join key (K_join)**: A symmetric join key used to join a secure target network. The value of key K_join is intended to be secret, and thus is intended to offer data confidentiality. The value of key K_join is updated during provisioning to a new value that is known only to the target network security manager and the device.[10]
- **Default join key value (K_global)**: A symmetric join key with a published value. The value of K_global is not intended to be a secret; its value is well known. It therefore does not offer data confidentiality, but does help improve data integrity. Its purpose is to establish connectivity between devices compliant to this standard that do not share a secret join key. Such connectivity is needed for:
  - over-the-air (OTA) provisioning of target network related information;
  - OTA reading of device identity and configuration settings;
  - OTA authentication of device credentials; and
  - OTA updating of join key K_join (the latter two steps using asymmetric cryptography).

  The value of the default join key shall be K_global, as defined in 7.2.2.2.
- **Open join key (K_join = K_open)**: A published non-secret value for the join key (K_join). This special value for the join key is used to join a provisioning network so that certain OTA symmetric-key only provisioning methods can be facilitated. The actual value for this key is defined in 7.2.2.2.
- **Physical and logical networks**: A physical network is a set of physical devices that communicate with each other, possibly through multiple hops. A logical network is a network instance that runs on the physical network. One physical network may support multiple logical networks. Logical networks have different individual priority and security properties. For example, the target network and the provisioning network are two logical networks that exist on a physical network.
- **Idle state**: Device state that is not actively participating in the wireless network,
- **Provisioning state**: The device is in the provisioning phase.
- **Provisioned state**: The device received enough information to join target network, and got the DMO.Join_Command=1.
- **Factory defaults**: The default configuration of a field device as it comes out of a manufacturing facility. The default configuration has K_global and K_join equal to their default values, and OTA provisioning allowed. An operational device may be reset to factory default, either by the system manager when it is part of a secured network or by OOB means using a provisioning device. Factory defaults for the provisioning process are summarized in Table 367. Only the system manager shall have the authority to reset a device to factory defaults via the network.

This specification does not preclude devices that do not allow reset to factory defaults.

---

10 Appropriate mechanisms are provided so that the protocol suite defined by this standard cannot be used to read the current value of the join key from a device. Note that the secrecy of join keys cannot be enforced by this standard.

13.3 Provisioning procedures

All field devices compliant with this standard shall implement a standard object called the device provisioning object (DPO). Attributes of the DPO in the DBP shall specify the information required to initiate a join request to the target network. The device shall retain all attributes of the DPO through a power cycle or battery replacement. The device provisioning object is described in detail in 13.9.1.

This specification does not preclude that the system manager can have the DPO, for example, store the security manager's EUI64Address.

PDs shall implement a device provisioning service object (DPSO) that contains information intended for the DBPs that are serviced by the PD.

Provisioning involves setting the attributes of the DPO. The attributes in the DPO contain both network-related and trust-related information. These attributes can be set via three different means:

- they may be pre-installed during device manufacture; or
- they may be set using OOB means; or
- they may be set by a PD using a provisioning network, where the PD acts as a proxy for a security manager/system manager of a target network to provide the trust-related and network-related information for that target network.

All devices complying with this standard shall support the formation of the provisioning network using only the full protocol suite defined by this standard (PhL, DL, NL, and TL), i.e., not requiring any other mechanism. However, this standard does not disallow provisioning by OOB communication means.

When using the Type A field medium (5.2.6.4) in the provisioning network, standard PDUs shall be used to set the attributes of the DPO, which defines a set of default read-only attributes for the formation of either the provisioning network or another unsecured network. The default attributes include published default symmetric keys (K_join = K_global and K_join = K_open), a default D-subnet identifier, and a default set of channels. Since this set of default attributes is known and contained in the DPO of all devices conforming to this standard, those attributes provide a means for all devices to join a provisioning network.

The DPO includes an attribute, DPO.Allow_Provisioning, that specifies whether access to the attributes of the object via the default open instance is either allowed or blocked.

Some devices may implement an external mechanism (i.e., a switch) that will lock the provisioning state (either blank or provisioned) of the device, to minimize battery consumption and also to minimize the likelihood that a rogue PD will re-provision a device.

13.4 Pre-installed symmetric keys

The formation of a provisioning network is not a necessary step for provisioning; the trust-related information can be pre-installed in a device. For example, it is possible for a user to delegate (partly) the provisioning of devices to a device manufacturer or to a third party. A device manufacturer may pre-program secret symmetric join keys into devices, and may supply this same secret symmetric join key data to the user so that the data can be loaded to the system/security manager of the target network. Alternatively, the user may stipulate to the device manufacturer what symmetric key shall be loaded. In this case, the DPO of a device shall be pre-installed with the target network information and the target symmetric join key K_join. Depending on the application, two or more devices may share the same secret information. Devices with pre-installed trust information and target network information can proceed directly to the subnet joining process.

When a device has pre-installed trust-related information but no target network-related information, it shall be possible to provision the device with necessary network information. This facilitates having the device receive advertisements from the target network on the intended channels, expediting the subnet joining process and present join requests only to the target network. If the network-related information is not provisioned, a device may use the default network settings to scan for advertisements from all networks in its vicinity (including those of competitors of the device's owning organization).

13.5 Provisioning using out-of-band mechanisms

Devices without pre-installed symmetric keys need to be authenticated and then provisioned with trust information. As noted earlier, this can be accomplished either through the provisioning network Type A field medium over-the-air or through OOB mechanisms.

OOB communication means include, but are not limited to, infrared, wired connectors, memory cards, keyboards on devices, NFC, and plugs. The mechanism of OOB communications is outside the scope of this standard. The attributes of the DPO that specify the joining to the target network should be set to the same values regardless of the means used (over-the-air or OOB).

13.6 Provisioning networks

13.6.1 General

In addition to OOB provisioning and factory pre-provisioning, this standard defines the formation of a standard network for provisioning devices over-the-air (OTA) using the Type A field medium. The default symmetric join key (K_global) or open symmetric join key (K_join = K_open) may be used as the trust-related information for the formation of the OTA provisioning network. The default join key (K_global) is used for the formation of the provisioning network to obtain target network-related information and target network join key and for devices with asymmetric cryptographic capability. The join key (K_join = K_open) is used for the formation of a provisioning network where both trust-related and network-related information is provisioned over-the-air. This form of provisioning is insecure and by default system managers and provisioning devices shall not allow joining with this join key.

A PD that has asymmetric cryptographic capabilities distinguishes the method with the key used to generate the MIC in the Security_Sym_Join().request. In the PD, the MIC generated by the device joining the default network needs to be validated a maximum of twice – one for K_open and the other for K_global. If the security manager detects that K_global is used for the MIC, the DBP shall be provisioned using asymmetric cryptography. Otherwise, the DBP shall be provisioned using the K_open symmetric key.

The provisioning network can either be an isolated mini-network formed with a handheld device, or it can be a separate logical network on the target network itself. In the latter case, connectivity from the DBP to the advertising router is open but connectivity further on, from that advertising router to the system manager, is protected by the existing session and thus secured. If the logical provisioning network is on the target network, the application objects of the system/security managers on the target network and the logical provisioning network (e.g., DPSO) can communicate with each other within the same device.

Figure 135 illustrates the provisioning (mini-)network.

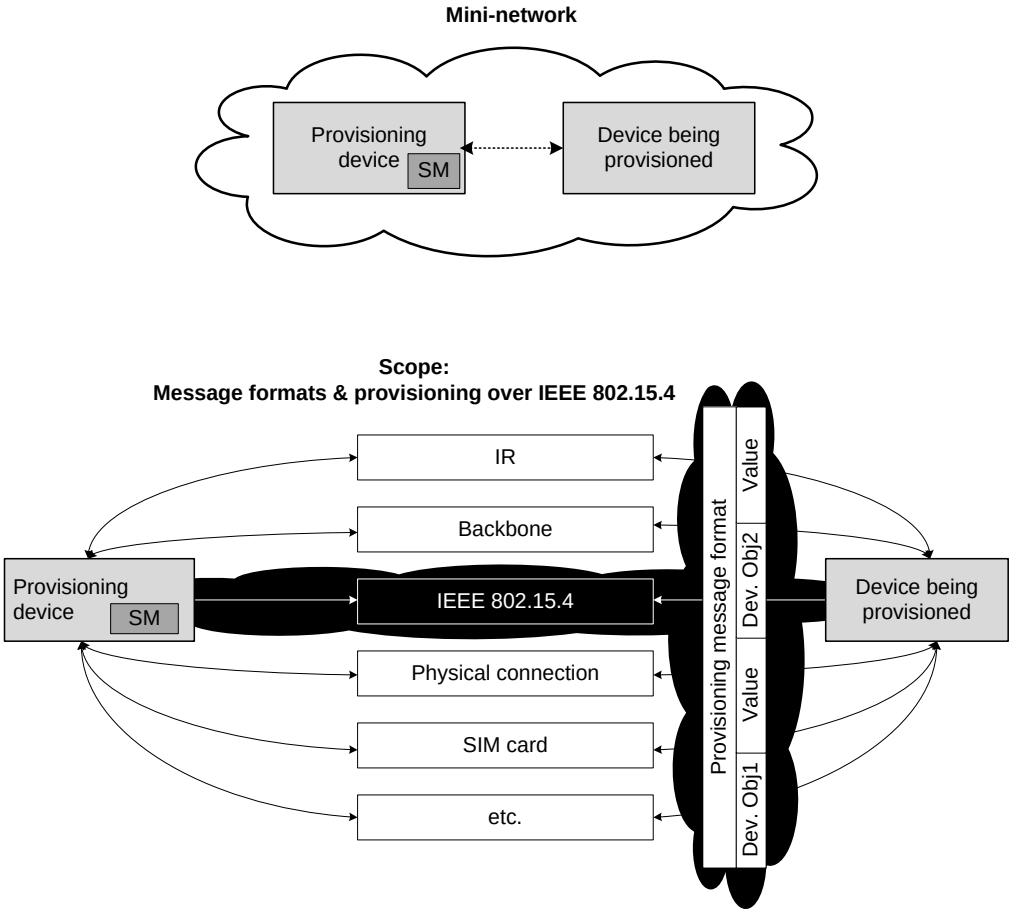


Figure 135 – The provisioning network

OTA provisioning uses a PD that can be either:

- a handheld configurator that forms an isolated mini-network with the DBP. This handheld has its own system/security manager and an advertising router functionality; or
- the system/security manager of the logical provisioning network on the target network.

NOTE When a PD is used for OTA provisioning, it forms a mini-network and functions temporarily as both the system manager and security manager for the DBP.

13.6.2 Provisioning over-the-air using asymmetric cryptography

DBPs that are capable of performing asymmetric cryptographic calculations shall use the default join key (K_global) to join a provisioning network. The DBP receives advertisements whose D-subnet ID = 1 from nearby advertising routers and initiates a join request using the default symmetric key. A successful subnet joining process results in the PD and the DBP having established a contract for further communication. The PD then uses standard AL primitives (such as read and write) to transfer the network-related information contained in its DPSO to the DPO of the DBP.

For provisioning the trust-related information the PD interrogates the DBP; i.e., it reads its credentials (e.g., DPO.PKI_Certificate or multiple DPO.PKI_Certificates; see Annex G), and sends those credentials to the security manager. The security manager of the provisioning network checks the credentials of the DBP and validates the authenticity of the DBP through a challenge-response mechanism. The security manager/system manager may ask for further confirmation from the user through a GUI to provision the DBP. Once accepted, the system manager provides the DBP with the secret join key, K_join, so that the DBP can join the target

network either immediately or at a later time, using that join key. When this new key is transmitted over-the-air, it shall be encrypted by the asymmetric key of the DBP, which is part of its certificate, so that it cannot be recovered by an eavesdropper while in transit.

Security managers conforming to this standard are not required to have asymmetric cryptographic capabilities; hence, some security managers may not be able to accept or provision devices using asymmetric cryptographic capabilities. When the DBP joins the provisioning network using K_global, security managers and PDs not capable of asymmetric cryptographic calculations shall not transmit the trust-related information to the DBP.

In addition to a high level of security, asymmetric cryptographic modules and certificates provide a convenient and easy means for devices to establish communication with the security manager of the target network and to be provisioned without the use of additional tools. It is recommended that manufacturers of security manager devices that lack support for asymmetric cryptography provide adequate means (e.g., memory, processing power, or optional peripherals, etc.) to upgrade such security managers, upon user request, to support asymmetric cryptography.

13.6.3 Provisioning over-the-air using an open symmetric join key

This standard allows PDs to provision devices that do not have asymmetric cryptographic capabilities to be provisioned over-the-air. For this purpose, a well-known open symmetric join key (K_join = K_open) is used. By default, the security manager in the PD shall not permit OTA provisioning with the open symmetric key, K_open. The provisioning network is not secure in itself, since it uses a published open key and join key for the target network, and thus requires compensating measures, such as a secure physical connection or use of asymmetric cryptography, for security.

NOTE 1 In OTA provisioning with K_global, the security information (i.e., join key) is encrypted with an asymmetric key while transmitting.

NOTE 2 The use of an open symmetric join key for provisioning is not a secure procedure. An eavesdropping device may be able to obtain the join keys to the target network and pose a security risk when this provisioning procedure is used. This provisioning procedure can only be used in applications where the security risk is minimal and the user is either not concerned or has taken sufficient measures to avoid eavesdropping. Such exposure can be avoided by using asymmetric crypto-based provisioning or OOB provisioning.

Use of the symmetric join key K_open for provisioning is a configuration option. DBPs may be pre-configured not to use OTA provisioning with this key. By default, security managers and PDs shall reject join requests from all devices that send join requests using the K_open symmetric join key. Security managers and PDs need to be configured to accept join requests using the K_open join key. It is permissible for security managers and PDs not to permit such configuration.

A device that joins a provisioning network using the K_open join key may be provisioned by the PD with the join key for a target network. However, once provisioned with a new join key for the target network, the device shall not be allowed to use the K_open symmetric join key unless the device is reset to factory defaults. Thus the only permissible means for the device to reuse this key for provisioning is to reset the device to factory defaults.

The provisioning procedure using the K_open symmetric join key can be used either in the provisioning (mini-)network or through a separate logical network on the target network. The DBP receives an advertisement from a provisioning network and a standard join request is sent using a symmetric key (K_join = K_open). If the request is accepted, the DBP joins the provisioning network and a contract is established between the DBP and PD. Application-level read/write primitives and methods are available to the PD to provision the trust-related and network-related information; this includes, for example, provisioning the target join key using the DPO.Write_Join_Key method.

The provisioning (mini-)network can also be used for device configuration. Since a contract has already been established, the PD may also use the network (either OTA or OOB) to configure the DBP.

13.7 State transition diagrams

The options discussed thus far for provisioning are shown below in state transition diagrams.

Figure 136 depicts the state transitions relevant for provisioning through the lifecycle of a field device. The diagram depicts states at a manufacturing site and a user site.

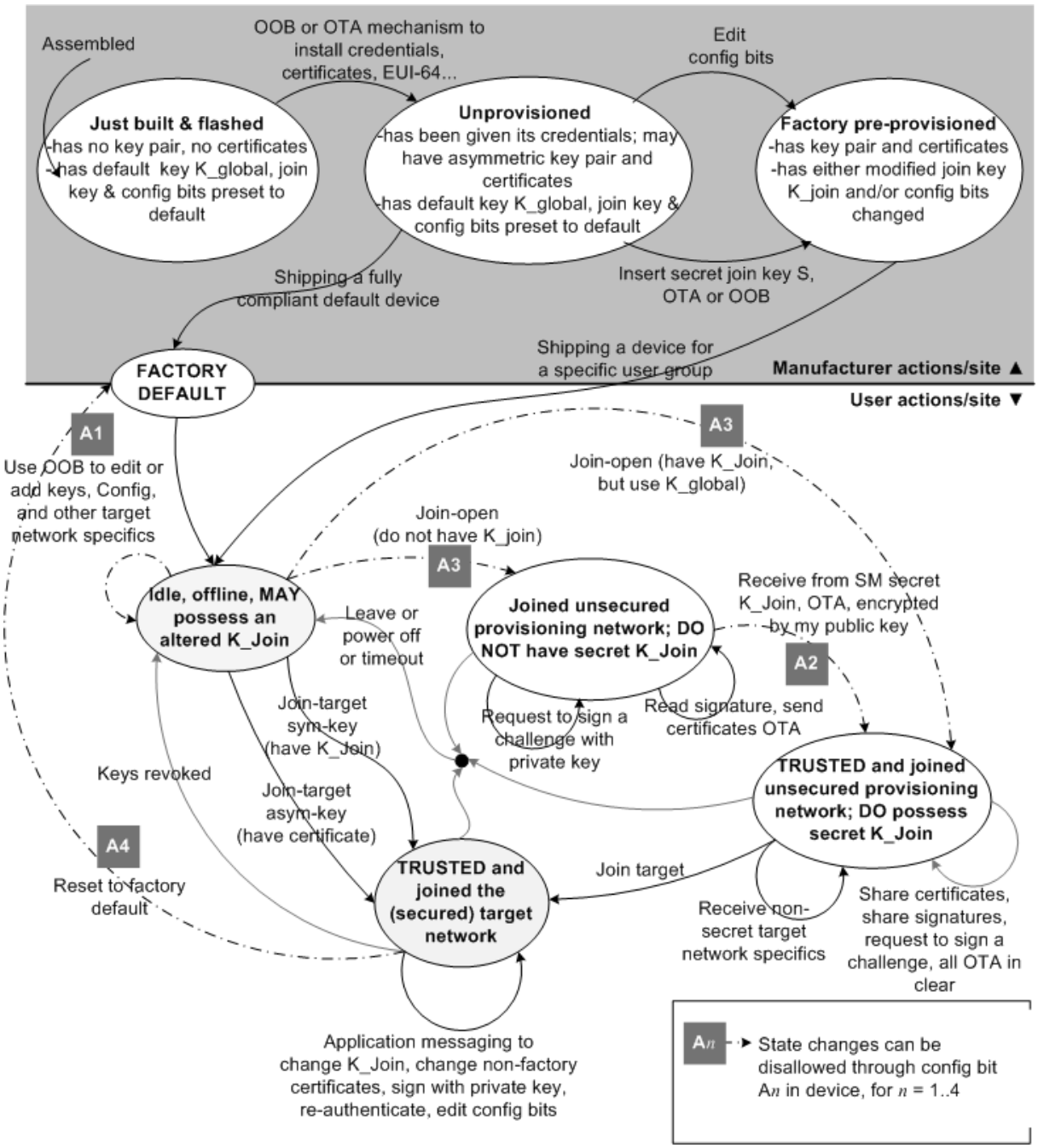


Figure 136 – State transition diagrams outlining provisioning steps during a device lifecycle

A field device that is newly manufactured transitions to the unprovisioned state when its identity (e.g., it EUI64Address) and credentials are provided to it. In this state, the device has the default settings as defined in Table 367.

Table 367 – Factory default settings

Attribute	Description	Default value
Default symmetric join key K_global	The symmetric join key used to join a default network	Specified in 7.2.2.2.
Open symmetric join key (K_join = K_open)	The symmetric join key used to join a provisioning network, then to receive new target join keys	Specified in 7.2.2.2
Allow OOB provisioning (A1)	This configuration bit allows the use of OOB mechanisms for provisioning the device. This bit is irrelevant if the device does not have any OOB means for provisioning	1 = allowed
Allow asymmetric-key-based provisioning (A2)	This configuration bit allows the use of asymmetric crypto for OTA provisioning of K_join. This bit is irrelevant if the device does not support asymmetric cryptography	1 = allowed
Allow default join (A3)	This configuration bit allows the device to join a default network. Some devices may choose not to allow a default join at all	1 = allowed
Allow reset to factory defaults (A4)	This configuration bit allows execution of OTA commands that reset the device to the factory default configuration	1 = allowed

The manufacturer may ship devices with these default settings.

Alternatively, the device may be pre-provisioned for a particular user at the manufacturing site. When a device is pre-provisioned, the default settings of the device are changed. The device may be given a new target symmetric join key specific to a target network at the user site. In addition, any of the configuration bits (A1, A2, A3 and A4) may be changed. For example, the default network join and reset to factory defaults may be disabled (A3, A4 = 0). Such a device shall not be able to be provisioned through the open symmetric join key (K_join = K_open).

The device arrives at the user site either pre-provisioned or with factory defaults and is in the idle state.

At a user site, a device may be provisioned, using OOB mechanisms (A1 enabled), with a symmetric join key and network-related information for joining the target network. Alternatively, the device may already have preinstalled secret join keys and/or network-related information established by the device manufacturer. If the network-related information is not provisioned into the device at manufacturing, the device may join a provisioning network using the default symmetric join key K_global, specified in 7.2.2.2 (if A3 is enabled), and may be provisioned with network-related information using over-the-air mechanisms.

Devices that fail to join the target network using their provisioned information may seek to join a provisioning network (if A3 is enabled) using K_global. After joining using the default join key (K_global), the PD may use the Write_Symmetric_Join_Key method to update the K_join only if it is sent encrypted with the asymmetric key of the DBP.

If the device in an idle state does not have a valid installed symmetric join key and is allowed to join a default network, and A4 is enabled, the device shall start scanning for advertisements in order to reach a security manager/system manager of a default network in its vicinity.

If an advertisement is found and the device has asymmetric cryptographic capabilities and PKI certificates, it shall forward its credentials to the security manager associated with the advertising router. The advertising routers shall forward join requests to their security

managers using an established contract that the advertising router has with the security manager/system manager.

When the security manager receives new device credentials, it first checks whether devices with those credentials are expected and authorized for the target network. This may be accomplished via lookup in pre-populated white lists with the EUI64Address of the individual device. The device credentials are used by the system manager to decide on the CA (and its asymmetric key) to use in subsequent authentication steps.

If the device is authorized, then the authenticity of the credentials is checked by the system manager. The device credentials include the device certificate or multiple certificates. When using multiple certificates, the check on the device data may[11] consist of two asymmetric crypto steps, one using the CA's public key that is already present inside the security manager (the PD) to read the first certificate (termed the issuer certificate) and hence the issuer's public key, followed by the second certificate (termed the device certificate) and hence the device's public key, using the issuer's public key. Once the device's public key is obtained, a challenge/response mechanism (see 7.4.6) is used by the PD to establish the authenticity of the DBP.

A copy of data exchanged in the preceding steps may be logged in public files in the PD for future audit purposes.

User input to accept the device may be solicited before the device is accepted. A dialog on a human-machine interface (HMI) connected to the system manager may seek confirmation that the trustworthy device should be allowed to join the target network. This can be a yes/no dialog that asks if a specific device, with a specific authenticated identity, that is a member of a family of expected and deemed welcome devices, should indeed now be prepared for a secure join to the secured target network. When this user-input step is implemented, and the user response is not received and no response is sent within the join response timeout period, the join request shall be considered to have failed.

If the device is authorized (present in the white list) and authentic, the PD generates a new key for the DBP, encrypts it using the DBP asymmetric key and transmits it to the DBP. A copy of that may be logged in public files in the PD for future audit purposes.

Failure in any of the steps above can be due to loss of connectivity, timeouts, or denial of join request from the DBP. Examples of the latter include a negative status on the white lists, a mismatch while authenticating, or a reject from a dialog on an HMI. When it is clear that a DBP should be rejected for any of those reasons, an alert is generated by the security manager. No join response shall be sent back to the device indicating a join failure to the device.

If the DBP does not have asymmetric cryptographic modules but has the open symmetric join key, it can join a provisioning network with the open symmetric join key (K_join = K_open). The right to accept or reject provisioning of DBPs that use the open symmetric join keys (K_join = K_open) rests with the PD. By default, the PD shall not provision devices that join with the open join key; however, the PD may be configured to provision such devices. If the PD is configured to allow open OTA provisioning, then the DBP will be provisioned with a new join key K_join for joining the target network. Once provisioned, the device shall not use the open key again unless it is reset to factory defaults (A4 is enabled).

Once provisioned, the device can proceed to join the target network with its provisioned information. As part of the subnet joining process, the device receives a master key, T-keys, and D-keys, in addition to establishing a contract with the system/security manager of the target network, and normal operation of the standard secured network follows.

---

[11] The two-certificate chain described here is only one of the many certificate topologies possible with multiple certificates. The DPO provides attributes to include multiple certificates.

As part of the normal operation of a network, the system manager of the network may provision the device with sufficient information to join another network when the device leaves the current network. This process enables a device to join and leave multiple networks. Provisioning for another network using a current target network is accomplished as follows.

a) The DPSO in the current system manager retrieves network information and security keys from the system/security manager of the other network.

NOTE The interface for such inter-manager communication is beyond this scope of this standard.

b) The DPSO in the current system manager installs information into the DPO of the device.

c) The DBP leaves the current network.

d) When the device leaves the current network, it joins the next network with network and security information installed in its DPO.

As described herein, there are multiple paths (and state transitions) available for an unprovisioned device to be provisioned and ultimately to join a secured network. These paths are illustrated via the state transition diagram in Figure 137. Figure 137 is related (and equivalent to) to Figure 136; however, Figure 137 is depicted from the perspective of a device internal state.

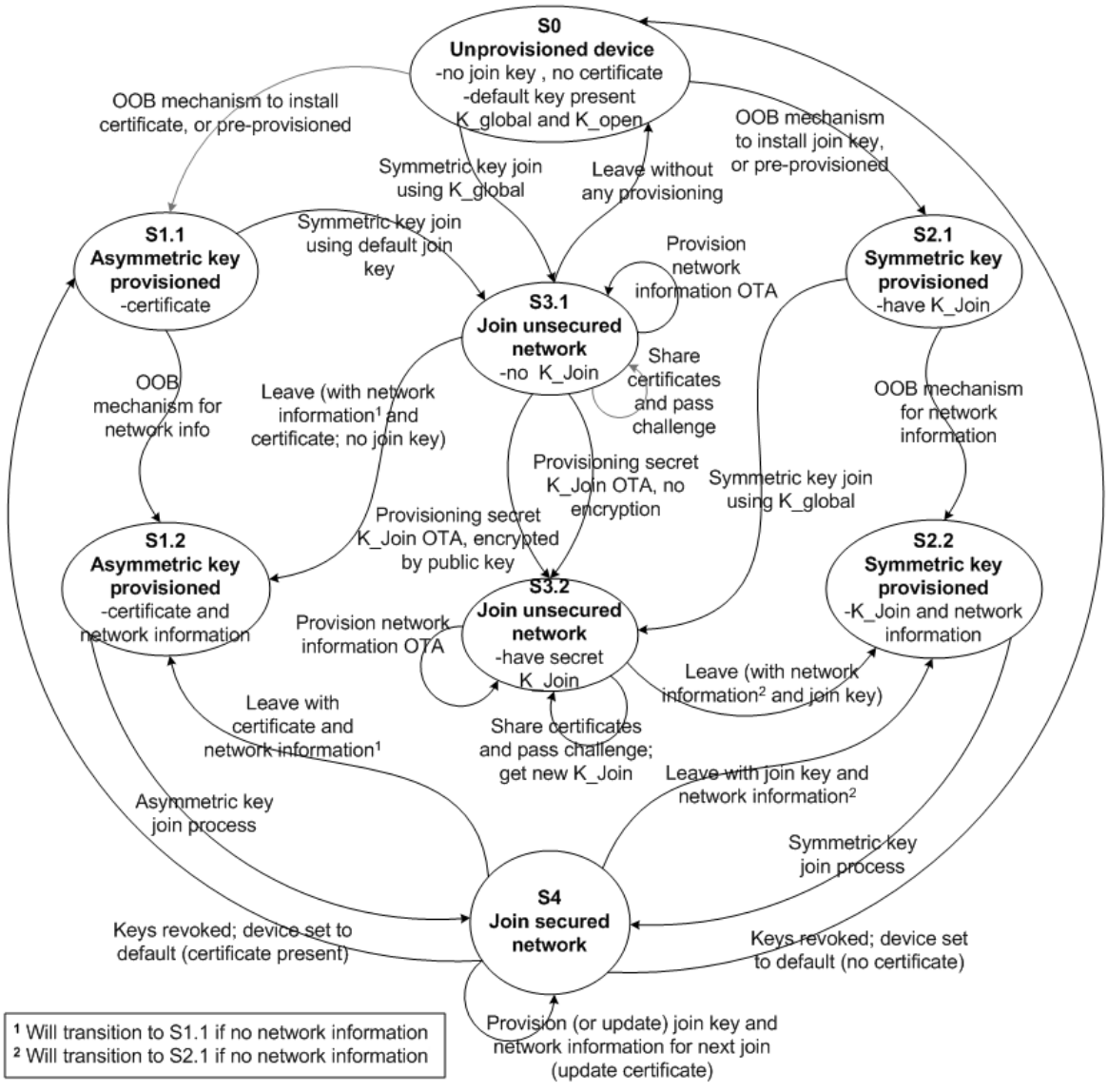


Figure 137 – State transition diagram showing various paths to joining a secured network

The transitions and paths addressed in Figure 137 include:

a) OOB provisioning of symmetric key and network information:

   1) State transitions : S0 → S2.1 → S2.2 → S4.

   2) Synopsis: OOB mechanisms are used to provision a device with the target network join key (S0 –> S2.1) and network information (S2.1 → S2.2). Then, the device uses the symmetric join procedure (S2.2 → S4) to join the secured network.

b) Factory pre-provisioned (OOB or otherwise): Asymmetric keys and certificates and OOB provisioning of network information:

   1) State transitions : S0 → S1.1 → S1.2 → S4.

   2) Synopsis: A device is factory pre-provisioned with asymmetric keys and certificates (S0 → S1.1). The device has the necessary information to initiate an asymmetric-key join procedure. However, it does not have enough network-related information. This information is provisioned using the OOB mechanism (S1.1 → S1.2). Then, the device uses the asymmetric join procedure to join the secured network (S1.2 → S4).

c) OOB provisioning of symmetric-key information and OTA provisioning of network information:

   1) State transitions : S0 → S2.1 → S3.2 → S2.2 → S4.

   2) Synopsis: A device is provisioned using OOB mechanism (or pre-provisioned) with the symmetric join key for the target network (S0 → S2.1). The device then joins a default provisioning network using the default join key, K_global (S2.1 → S3.2). The PD in the provisioning network provides the network information for the target network. The device leaves the provisioning network (S3.2 → S2.2) and joins the secured network (S2.2 → S4) using the symmetric join procedure.

d) Factory pre-provisioned (OOB or otherwise) asymmetric keys and certificates and OTA provisioning of symmetric keys:

   1) State transitions: S0 → S1.1 → S3.1 → S3.2 → S2.2 → S4.

   2) Synopsis: A device is factory pre-provisioned with asymmetric keys and certificates (S0 → S1.1). The device has the necessary information to initiate an asymmetric-key join procedure. However, it cannot join a target network that does not support an asymmetric subnet joining process. The device then joins a default provisioning network that is different from the target network using the default join key, K_global (S1.1 → S3.1). As part of this provisioning network, the device exchanges its credentials, passes a challenge-response mechanism, and receives the target network join key, encrypted with the device's public key, from the PD (S3.1 → S3.2). The device is then provisioned with the network information OTA. Then, the device leaves the provisioning network (S3.2 → S2.2) and joins the secured network (S2.2 → S4) using the symmetric join procedure.

e) Open join key-based provisioning in the clear:

   1) State transitions : S0 → S2.1 → S2.2 → S4(1) → S2.2 → S4(2).

   2) Synopsis: A device that has the default open symmetric join key. It uses the symmetric join key procedure for joining a provisioning network (S2.2. → S4(1)). As part of this provisioning network, the device is provisioned with the target network join key and network information. The device then leaves the provisioning network (S4(1) → S2.2). The device is now provisioned to join the target network; it joins the secured target network using the symmetric-key subnet joining process (S2.2 → S4(2)). In this transition, the first time the device has joined a provisioning network is indicated by state S4(1), and the second time it is joined to the target network is indicated by state S4(2). After the device has reached S4(2), the device cannot use the open symmetric join key unless it is reset to factory defaults.

13.8 Device management application protocol objects used during provisioning

This standard uses one DMAP object and one SMAP object during provisioning. The device provisioning object (DPO) holds the configuration settings. Figure 138 illustrates provisioning objects and the interactions between them.

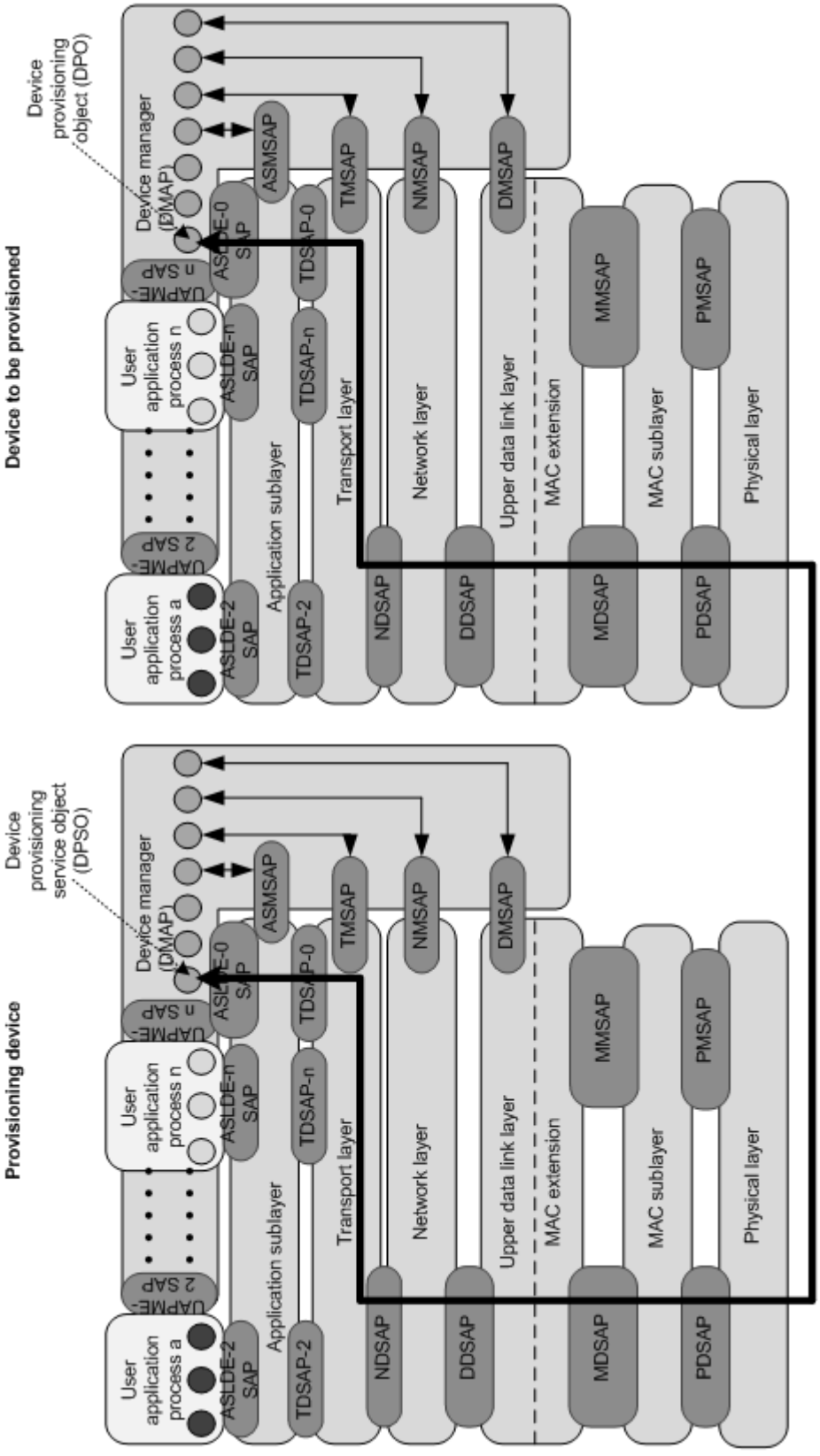


Figure 138 – Provisioning objects and interactions

Whether it is the system manager/security manager in a handheld device or the system manager of the target network, the PD shall implement a device provisioning service object (DPSO) with attributes and methods to provision the DBP. The DPSO may have a list of symmetric keys, used to provision devices that do not have pre-installed keys.

The white list, symmetric-key information, and target network information in the DPSO can be maintained with information specific to a device in the White_List_Array attribute in Table 372. Alternatively, a pool of valid symmetric keys can be maintained.

When the DBP joins the provisioning network using K_global, a contract is established between the PD and DBP. The DPSO in the PD can use the established contract to communicate with the DPO in the DBP. Read and write primitives are used for accessing and setting the attributes of the DPO. A subset of network and trust information can now be provisioned in the DPO using the DPSO. To write the new symmetric join key to the device the DPSO invokes the Write_Join_key method of the DPO. This method is allowed if the new key value was received under protection of asymmetric crypto. The attributes in the DPO include both network-related and trust-related information.

Users that want additional security while provisioning should use the asymmetric crypto-based authentication and secured key loading technique for the trust-related steps. Alternatively, out-of-band mechanisms may be used for provisioning join keys.

Once the appropriate trust and network information has been provisioned in the DPO, the device is ready to join the target network. The provisioning network can also be used for device configuration. Since a contract has already been established, the PD may also use the network (or OOB means) to configure the appropriate UAP and DMAP objects of the DBP.

The device provisioning object (DPO) provides an attribute called the Target_DL_Config in the DL_Config_Info format. DL_Config_Info is described in Clause 9 (see Table 102) to configure various attributes of the DL. Once provisioned with this attribute, the DPO provides the DL with an OctetString encapsulating DL_Config_Info that includes at least one superframe, and at least one link, that can be used by the DL in searching for advertisements. Target network-specific (e.g., non-default) timeslot templates, channel-hopping patterns, superframes and links can also be provided to the DBP through the Target_DL_Config attribute. Such configuration helps reduce the amount of information (e.g., join superframes) that is otherwise required to be advertised by target network advertisement routers.

The DL of the device plays a major role during the provisioning and joining of the device. The state machine of the DL when it is going through the provisioning process is described in 9.1.14.2.

If the provisioning process is successful, the DPO provides the DL with the set of attributes, including D-subnet information, superframes, and links, that the DL can use to search for the target network and corresponding D-subnet(s). In the provisioned state the DL operates its state machine as configured in the superframes and links that were provided by the DPO. Superframe operation may be delayed or disabled by setting the IdleTimer field within the superframe.

Since the device retains the information that was used to provision the DL (all attributes of the DPO), this ensures a means to reset the DL back to its provisioned state by putting the DL into its default state and then adding the provisioned attributes.

The DPO shall be accessible to the system manager of the target network after joining with Key_Join. Once the device joins a target network, the system manager of the target network has the ability to change the attributes of the DPO. The system manager of the target network has the ability to instruct the device to join another target network by providing network and trust information of the other network. Depending on the value of configuration bit A4, the

system manager of the target network has the ability to invoke a DPO.Reset_To_Defaults method to remove trust information from the device.

NOTE In the provisioning phase, the DPO in the DBP is accessed by the system/security manager functionality in the PD.

13.9 Management objects

13.9.1 Device provisioning object

Table 368 describes the attributes of the DPO. The data type, default value, and a brief description are provided for each attribute. Each attribute also has accessibility of read only or read/write. The attributes of the DPO are accessible only to the system/security manager. The value of a read-only attribute can be set only at the device manufacturing time (i.e., at a time before the device is certified) or internally by the device; no entity external to the device can change this attribute. Read/write accessibility implies that entities external to the device can change the value of the attribute. The attributes of the DPSO are accessible (read/write) only to the system manager.

The attributes classified as “constant” have a value that is not changed during the device lifecycle, neither internally nor externally. The definition of the classification is found in 12.6.3.

Table 368 – Device provisioning object *(1 of 6)*

Standard object type name: Device provisioning object (DPO)				
Standard object type identifier: 120				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Default_NWK_ID	1	A published network identification for the default network	Type: Unsigned16 Classification: Constant Accessibility: Read only Default value: 0x0001	This is the network identification for the default network. The default network may be used to form the provisioning mini-network
Default_SYM_Join_Key	2	A published join key for the default network	Type: SymmetricKey Classification: Constant Accessibility: Read only Default value: K_global, 7.2.2.2	This key is used by devices to join the default network. The default keys may be used to form the provisioning mini-network
Open_SYM_Join_Key	3	A published join key for the default network	Type: SymmetricKey Classification: Constant Accessibility: Read only Default value: K_open, 7.2.2.2	This key is used by devices to join the unsecured provisioning network

Table 368 *(2 of 6)*

Standard object type name: Device provisioning object (DPO)				
Standard object type identifier: 120				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Default_Channel_List	4	The list of 2,4 GHz channels used by the default network. The attribute is coded as a bit map of 16 bits representing the 16 frequencies	Type: Unsigned16 Classification: Constant Accessibility: Read only Default value: 0x7FFF	The list of channels used by the advertising routers of the default network. To join the default network the device may receive advertisements on any of these frequencies
Join_Method_Capability	5	The join capabilities of the device	Type: Unsigned2 Classification: Constant Accessibility: Read/write Default value: 00 Named values: 00: default join only; 01: symmetric-key join only; 10: asymmetric-key join only; 11: any key join	This attribute defines the capability of a device to join. The device can either use symmetric keys or asymmetric-key infrastructure to join a target network. This attribute merely defines the capabilities of the device. The actual method used to join the target network shall be set by the PD
Allow_Provisioning	6	A Boolean value set to indicate if a device is allowed to be provisioned or not	Type: Boolean1 Classification: Static Accessibility: Read/write Default value: TRUE	This flag is used to lock the state of an already provisioned device. If this value is set the device will not accept any reads or writes to the target network attributes
Allow_Over_The_Air_ Provisioning	7	A Boolean value set to indicate if a device is allowed to be provisioned or not	Type: Boolean1 Classification: Static Accessibility: Read/write Default value: TRUE	This Boolean indicates whether over-the-air provisioning is enabled or disabled. If over-the-air provisioning is disabled the device needs to be provisioned using out of band methods. Backbone devices shall have this value set to FALSE. In all cases, provisioning is allowed only if the Allow_Provisioning attribute is enabled

Table 368 *(3 of 6)*

Standard object type name: Device provisioning object (DPO)				
Standard object type identifier: 120				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Allow_OOB_Provisioning	8	A Boolean value set to indicate if a device is allowed to be provisioned using OOB means	Type: Boolean1 Classification: Static Accessibility: Read/write Default value: TRUE	The Boolean is used to block the devices from accepting provisioning information from OOB means
Allow_Reset_to_Factory_ Defaults	9	A Boolean value set to indicate if a device is allowed to be reset to factory defaults	Type: Boolean1 Classification: Static Accessibility: Read/write Default value: TRUE	This Boolean is used to block devices from being reset to factory defaults by a system manager
Allow_Default_Join	10	A Boolean value set to indicate if a device is allowed to join a network using the default keys	Type: Boolean1 Classification: Static Accessibility: Read/write Default value: TRUE	The Boolean is used to force the devices to join a particular target network and not join to any default network. Devices choosing not to join a Default network can set this attribute to FALSE
Target_NWK_ID	11	The network ID of the target network that this device is provisioned to join	Type: Unsigned16 Classification: Static Accessibility: Read/write Default value: 0	This attribute indicates the target network that this device has to join [a]
Target_NWK_BitMask	12	A bit mask for matching of the bits of the Target network ID	Type: Unsigned16 Classification: Static Accessibility: Read/write Default value: 0xFFFF	The bit mask is useful for matching multiple target networks. If the value of a bit in the bit mask is 1 then the bit has to be exactly matched to the corresponding bit in the Target Network. The default value of all 1s indicates that all bits of network ID need to match

Table 368 *(4 of 6)*

Standard object type name: Device provisioning object (DPO)				
Standard object type identifier: 120				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Target_Join_Method	13	Indicates whether the device should use symmetric-key join or asymmetric-key join mechanism to join the target network	Type: Unsigned1 Classification: Static Accessibility: Read/write Default value: 1 Named values: 0: symmetric key 1: asymmetric key	Subclause 7.4.4 defines two different methods for join depending on the use of either symmetric keys or asymmetric-key certificates. This attribute sets the method to be used to join a target network
Target_Security_Manager_EUI	14	The EUI64Address of the security manager in the target network that the device is intended to join	Type: EUI64Address Classification: Static Accessibility: Read/write Default value: 0xFF…FF (all 0xFF)	Set to the EUI64Address of the security manager that the device is provisioned to join
Target_System_Manager_Address	15	The IPv6Address of the system/security manager in the target network that the device is intended to join	Type: IPv6Address Classification: Static Accessibility: Read/write	The IPv6Address is required for backbone devices to join the network. The backbone devices do not have an advertising router – hence a join request is sent to the IPv6Address of the system/security manager to begin the subnet joining process. I/O devices and routing devices need not be provisioned with this attribute
Target_Channel_List	16	The list of channels used by the target network. The attribute is coded as a bit map of 16 bits representing the 16 frequencies	Type: BitArray16 Classification: Static Accessibility: Read/write	The target network may be using only a subset of channels for advertisements by the join routers. By using only a subset of frequencies battery powered devices can quickly join the target network by listening in that subset of frequencies only

Table 368 *(5 of 6)*

Standard object type name: Device provisioning object (DPO)				
Standard object type identifier: 120				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Target_DL_Config	17	The DL configuration information for this device	Type: OctetString Classification: Static Accessibility: Read / Write	This attribute indicates the various configuration settings for the DL of the device. The structure of this attribute is defined in DL_Config_Info defined in Clause 9
PKI_Certificate_Type	18	(Asymmetric-crypto option) The type of certificate stores in PKI_Root_Certificate and PKI_Certificates	Type: Unsigned8 Classification: Static Accessibility: Read/Write Default value: 0 Named values: 0: implicit cert; 1: manual cert	This field indicates a type of Certificate in PKI_Root_ Certificate and PKI_Certificates
PKI_Root_Certificate	19	(Asymmetric-crypto option) The root certificate of the certificate authority issuing the certificate to the device	Type: OctetString Classification: Static Accessibility: Read/write	The root certificate of the certificate authority and its corresponding asymmetric key is used to verify certificates of the peer nodes. The root certificate may be updated by the system manager
Number of PKI_Certificates	20	(Asymmetric-crypto option) The number of certificates stored in the PKI_certificate attribute	Type: Unsigned8 Classification: Static Accessibility: Read/write Default value: 0	This field indicates the number of certificates available in attribute PKI_Certificate
PKI_Certificate	21	(Asymmetric-crypto option) The certificate issued to this device for joining using the asymmetric-key infrastructure	Type: Array of OctetString Classification: Static Accessibility: Read/write	If Target_Join_ Method is set to Asymmetric-key, this attribute contains the certificate (which includes the asymmetric key, device ID, and other text) signed by a certificate authority which is required for joining the target network

Table 368 *(6 of 6)*

Standard object type name: Device provisioning object (DPO)				
Standard object type identifier: 120				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Current_UTC_Adjustment	22	The current value of the UTC accumulated leap second adjustment	Type: Integer16	See Table 25, attribute 1 and footnote
			Classification: Static	
			Accessibility: Read/write	
			Default value: 35	
[a] If the Target_NWK_BitMask (attribute 12) is set to 0xFFFF, the device shall ignore advertisements from routers belonging to any other network except the indicated target network. Otherwise a combination of network ID and bit mask shall be used. (See description of attribute 12 on how the NetworkID and bitmask are combined). This helps with fast joins and also prevents devices from trying to join all networks in their vicinity. This value can be set to 0 to allow responses to any advertising router.				

13.9.2 Device provisioning object methods and alerts

Several methods and alerts are available in the DPO. Table 369 describes the Reset_To_Default method.

Table 369 – Reset_To_Default method

Standard object type name(s): Device provisioning object (DPO)				
Standard object type identifier: 120				
Method name	**Method ID**	**Method description :**		
Reset_To_Default	1	This method is used to reset to default settings for the provisioning. This method shall be executed only when Allow_Provisioning is enabled		
	Input arguments (none)			
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Status	Unsigned8	Named values: 0: success; other: failure

Table 370 describes the method to write a symmetric join key. The join key shall not be exposed to a remote device, and may be exposed to limited internal process; DPO.Write_Symmetric_Join_Key() method installs a join key to a memory area that is not used for attributes (e.g., secure storage).

Table 370 – Write symmetric join key method

<table>
<tr><th colspan="5">Standard object type name(s): Device provisioning object</th></tr>
<tr><th colspan="5">Standard object type identifier: 120</th></tr>
<tr><th>Method name</th><th>Method ID</th><th colspan="3">Method description :</th></tr>
<tr><td rowspan="11">Write_SYM_join_key</td><td>2</td><td colspan="3">This method is used to write a symmetric join key to a device. This method is evoked by the DPSO to provision a DBP with the target join key. Depending on the provisioning method used this method call APDU and hence the join key may be encrypted by the T-key between the device and PD alone or the device’s asymmetric key in the APDU in addition to the APDU being encrypted by the T-key. This method shall be executed only when Allow_Provisioning is enabled</td></tr>
<tr><td colspan="4">Input arguments</td></tr>
<tr><td>Argument number</td><td>Argument name</td><td>Argument type (data type and size)</td><td>Argument description</td></tr>
<tr><td>1</td><td>New_Key_Value</td><td>SymmetricKey</td><td>New join key to be installed.</td></tr>
<tr><td>2</td><td>Encrypted By</td><td>Unsigned8</td><td>Named values:
0: TL_Session_Key_Only,
1: Asymmetric_Key</td></tr>
<tr><td colspan="4">Output arguments</td></tr>
<tr><td>Argument number</td><td>Argument name</td><td>Argument type (data type and size)</td><td>Argument description</td></tr>
<tr><td>1</td><td>Status</td><td>Unsigned8</td><td>Named values:
0: success;
> 0: failure</td></tr>
</table>

13.10 Device provisioning service object

13.10.1 Device provisioning service object attributes

Table 371 describes the attributes of the DPSO.

The system manager can either choose particular provisioning information for each EUI64Address or a set of join keys for a set of EUI64Addresses with no one-to-one mapping. The Boolean1 attribute, DPSO.Enable_White_List_Array, is used to specify which method is used.

In the DPSO.Enable_White_List_Array set, DPSO.White_List_Array is used to install particular provisioning information for each EUI64Address of the DBP.

If DPSO.Enable_White_List_Array is not set, the PD shall check if there are at least as many entries in DPSO.SYM_Key_List as entries in DPSO.White_List. This standard does not specify how each entry in DPSO.SYM_Key_List and DPSO.White_List is mapped.

Table 371 – Device provisioning service object *(1 of 4)*

Standard object type name: Device provisioning service object (DPSO)				
Standard object type identifier: 106				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
White_List	1	A list of devices permitted to be provisioned by this object	Type: Array of EUI64Address Classification: Static Accessibility: Read/write Default value: [ ] -- empty	This list contains EUI64Addresses of the device being provisioned. This list can be used to restrict a provisioning device to the specific set of devices whose EUI64Addresses are in this list. If this list is empty, then the provisioning device can provision any device
Symmetric_Key_List	2	A list of valid join keys with which a device can be provisioned	Type: Array of SymmetricKey Classification: Static Accessibility: Read/write Default value: {K_global] -- 7.2.2.2	This key is used by devices to join the target network, that have suitable entropy
Symmetric_Key_Expiry_Times	3	The expiration time for each key	Type: Array of TAITimeRounded Classification: Static Accessibility: Read/write Default value: [0xFFFF FFFF] [a]	This attribute sets the expiry time for each of the symmetric keys. The key is only used for provisioning if it has not expired
Target_NWK_ID	4	The network ID of the target network that the devices provisioned by this object are supposed to join	Type: Unsigned16 Classification: Static Accessibility: Read/write Default value: 0	This attribute indicates the target network (subnet ID) that a provisioned device has to join

Table 371 *(2 of 4)*

Standard object type name: Device provisioning service object (DPSO)				
Standard object type identifier: 106				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Target_Join_Method	5	A Boolean value to indicate if the devices provisioned by this object should use symmetric-key join or asymmetric-key join mechanism to join the target network	Type: Unsigned8 Classification: Static Accessibility: Read/write Default value: 0	Clause 7 defines two different methods for join depending on the use of either symmetric or asymmetric keys. This attribute sets the method to be used to join a target network. Named values: 0: symmetric key; 1: asymmetric key
Target_Security_Manager_EUI	6	The EUI64Address of the security manager in the target network that the device provisioned by this object is intended to join	Type: EUI64Address Classification: Static Accessibility: Read/write	Set to the EUI64Address of the security manager that the device is provisioned to join
Target_System_Manager_ Address	7	The IPv6Address of the system/ security manager in the target network that the device provisioned by this object is intended to join	Type: IPv6Address Classification: Static Accessibility: Read/write Valid range: all with highest bit reset	The IPv6Address is required for backbone devices to join the network
Target_Channel_List	8	The list of channels used by the default network. The attribute is coded as a bit map of 16 bits representing the 16 frequencies	Type: BitArray16 Classification: Static Accessibility: Read/write	The target network may be using only a subset of channels for advertisements by the join routers
Target_DL_Config	9	The DL configuration information for the device to be provisioned by this object	Type: OctetString Classification: Static Accessibility: Read / Write	This attribute indicates the various configuration settings for the DL of the device. The structure of this attribute is defined in the DL_Config_Info defined in Clause 9, Table 102

Table 371 *(3 of 4)*

Standard object type name: Device provisioning service object (DPSO)				
Standard object type identifier: 106				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
Allow_Provisioning	10	A Boolean value set to indicate if a device is allowed to be provisioned again or not	Type: Boolean1 Classification: Static Accessibility: Read/write Default value: TRUE	This flag is used to lock the future state of a provisioned device
Allow_Default_Join	11	A Boolean value set to indicate if a device provisioned by this object is allowed to join a network using the default keys	Type: Boolean1 Classification: Static Accessibility: Read/write Default value: Not_allowed (0)	The flag is used to force the provisioned devices to join a particular target network and not join to a default network. Once provisioned the device should join the target network
Enable_White_List_Array	12	A Boolean value set to indicate if the provisioning object is designed to set device specific provisioning information	Type: Boolean1 Classification: Static Accessibility: Read/write Default value: FALSE	If this flag is set the DPSO is capable of provisioning different devices (based on the EUI64Address) with different provisioning information. This can be used to provision a particular device, with a particular security manager and a particular Target_Join_Time, for example
White_List_Array	13	An array of the EUI addresses that the DPSO intends to provision along with the corresponding provisioning information for that device	Type: Array of DPSOWhiteListTbl Classification: Static Accessibility: Read/write	This attribute shall be used only if Device_specific_provisioning_flag is set. It contains the device specific provisioning information like device specific join keys, device specific target security manager, etc.
White_List_Array_Meta	14	Metadata for White List Array (Attribute 13) or set of White_List (Attribute 1), SYM_Key_List (Attribute 2)	Type: Metadata_attribute Classification: Static Accessibility: Read only	Metadata containing a count of the number of entries and capacity (the total number of rows allowed) of White_List_Array table or set of White_List [b]

Table 371 *(4 of 4)*

Standard object type name: Device provisioning service object (DPSO)				
Standard object type identifier: 106				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
DPSO_Alerts_AlertDescriptor	15	Used to change the priority of DPSO alerts that belong to the security category; these events can also be turned on or turned off	Type : Alert report descriptor Classification: Static Accessibility: Read/write Default value: [FALSE, 6]	See description of alerts in Table 374 and Table 375
Current_UTC_Adjustment	16	The current value of the UTC accumulated leap second adjustment	Type: Integer16 Classification: Static Accessibility: Read/write Default value: 35	See Table 25, attribute 1 and footnote

[a] When a computed key expiry time results in the value 0xFFFF FFFF, it is increased (circularly) to the modulo value 0x0000 0000, so that the value 0xFFFF FFFF can be used to designate a key that never expires.

[b] If Enable_White_List_Array is enabled, this attribute specifies a count of the number of entries and capacity for White_List_Array. If Enable_White_List_Array is disabled, this attribute specifies a count of the number of entries and capacity for the set of White_List and SYM_Key_List.

13.10.2 Device provisioning service object structured attributes

Table 372 describes the structured attributes of the DPSO. The White_List_Array is used when the PD has device-specific information, i.e., if the PD has symmetric join keys and other DPO attributes that are specific to a device. In this case, a structured array is required that stores the provisioning information indexed by the EUI64Address identifier of the DBP. This indexed array is described in Table 372.

After the PD receives the Device_SYM_Key from the security manager, the PD shall not expose the Device_SYM_Key attribute externally.

NOTE The interface between the PD and the security manager is beyond the scope of this standard.

Table 372 – DPSOWhiteListTbl data structure *(1 of 2)*

Standard data type name: DPSOWhiteListTbl		
Standard data type code: 440		
Element name	**Element identifier**	**Element scalar type**
Device_EUI	1	Type: Array of EUI64Address
		Classification: Static
		Accessibility: Read only
		Default value: [ ] -- empty
Device_Tag	2	Type: Array of VisibleString
		Classification: Static
		Accessibility: Read/write
		Default value: [“”]
Symmetric_Key_List	3	Type: Array of SymmetricKey
		Classification: Static
		Accessibility: Read only
		Default value: {K_global] -- 7.2.2.2
Symmetric_Key_Expiry_Times	4	Type: Array of TAIRounded
		Classification: Static
		Accessibility: Read only
		Default value: [0xFFFF FFFF] [a]
Target_NWK_ID	5	Type: Unsigned16
		Classification: Static
		Accessibility: Read only
		Default value: 0
Target_Join_Method	6	Type: Unsigned8
		Classification: Static
		Accessibility: Read only
		Default value: 1
		Named values: 0: symmetric key; 1: asymmetric key
Target_Security_Manager_EUI	7	Type: EUI64Address
		Classification: Static
		Accessibility: Read only
Target_System_Manager_Address	8	Type: IPv6Address
		Classification: Static
		Accessibility: Read only
		Valid range: all with highest-bit reset
Target_Channel_List	9	Type: Array of Unsigned8
		Classification: Static
		Accessibility: Read only

Table 372 *(2 of 2)*

Standard data type name: DPSOWhiteListTbl		
Standard data type code: 440		
Element name	**Element identifier**	**Element scalar type**
Target_DL_Config	10	Type: OctetString (See DL_Config_Info for format)
		Classification: Static
		Accessibility: Read only
Allow_Provisioning	11	Type: Boolean1
		Classification: Static
		Accessibility: Read/write
		Default value: TRUE
Allow_Default_Join	12	Type: Boolean1
		Classification: Static
		Accessibility: Read/write
		Default value: TRUE
[a] When a computed key expiry time results in the value 0xFFFF FFFF, it is increased (circularly) to the modulo value 0x0000 0000, so that the value 0xFFFF FFFF can be used to designate a key that never expires.		

When not null, the Device_Tag specifies a Tag_Name assigned to the device by a user. This value shall be written to the Tag_Name attribute of the DMO (see 6.2.8).

13.10.3 Device provisioning service object methods

Several methods are available for manipulating the DPSO. Standard methods such as read and write can be used for scalar or structured MIBs (SMIBs) in their entirety. The methods described herein are used to manipulate tables. These methods allow access to a particular row of a SMIB based on a unique key field.

It is assumed that the tables have a unique key field, which may either be a single element or the concatenation of multiple elements. The key field is assumed to be the (concatenation of) the first (few) element(s) of the table.

Table 373 describes the methods for manipulation of structured MIBs. These methods are based on the Read_Row, Write_Row and Delete_Row templates defined in Annex J.

Table 373 – Array manipulation table

Standard object type name(s): Device provisioning service object (DPSO)		
Standard object type identifier: 106		
Method name	**Method ID**	**Method description**
Setrow_WhiteListTbl	1	Method to set (either write or edit) the value of a single row of the white list array. The method uses the Write_Row method template defined in Annex J with the following arguments: Attribute_ID :14 (White_List_Array) Index 1: 1 (Device_EUI)
Getrow_WhiteListTbl	2	Method to get the value of a single row of the white list array. The method uses the Read_Row method template defined in Annex J with the following arguments: Attribute_ID :14 (White_List_Array) Index 1: 1 (Device_EUI)
Deleterow_WhiteListTbl	3	Method to delete the value of a single row of the white list array. The method uses the Delete_Row method template defined in Annex J with the following arguments: Attribute_ID :14 (White_List_Array) Index 1: 1 (Device_EUI)

13.10.4 Device provisioning service object alerts

Table 374 describes an alert to indicate a join attempt by a device that is not on the white list.

Table 374 – DPSO alert to indicate join by a device not on the WhiteList

Standard object type name(s): Device provisioning service object (DPSO)					
Standard object type identifier: 106					
Description of the alert: Alert to indicate provisioning request by a device not on white list					
Alert class (Enumerated: alarm or event)	**Alert category (Enumerated: device diagnostic, comm. diagnostic, security, or process)**	**Alert type (Enumerated: based on alert category)**	**Alert priority (Enumerated: high, med, low, journal only)**	**Value data type**	**Description of value included with alert**
0 = Event	2 = Security	0 = Not_On_ Whitelist_Alert	6 = Medium	Type: EUI64Address	EUI64Address of a device not on the white list

Table 375 describes an alert to indicate that inadequate capability is available for a device to join the network.

Table 375 – DPSO alert to indicate inadequate device join capability

Standard object type name(s): Device provisioning service object (DPSO)					
Standard object type identifier: 106					
Description of the alert: Alert to indicate inadequate device join capability					
Alert class (Enumerated: alarm or event)	**Alert category (Enumerated: device diagnostic, comm. diagnostic, security, or process)**	**Alert type (Enumerated: based on alert category)**	**Alert priority (Enumerated: high, med, low, journal only)**	**Value data type**	**Description of value included with alert**
0 = Event	2 = Security	1 = Inadequate_Join_ Capability_Alert	6 = Medium	Type: struct { reason: Unsigned8; rejectedDevice: EUI64Address}	The reason field provides a diagnostic code. Named values: 1: bad join key; 2: expired join key; 3: authentication failed The rejectedDevice field specifies the EUI64Address of the device that attempted to join

13.10.5 Summary of attributes that can be provisioned

The following is a summary of the attributes that are provisioned by the PD so that a new device can join a target network. These attributes can be provisioned either using over-the-air (OTA) methods or OOB methods. The list of provisioned attributes includes the follow items. The full list is defined by Table 368, Table 371, and Table 372.

- Trust-related information :
  - symmetric join key (K_join);
  - EUI64Address of the security manager;
  - network join method.
- Network-related information:
  - network ID and bitmask;
  - IPv6Address of the system manager;
  - DL configuration (contains the superframes, link TsTemplate, channel information, etc., needed to join the target network).

In addition, the configuration bits (attributes 6..10 of the DPO), describing the behavior of the device, can be set by the provisioning device.

13.11 Provisioning functions (informative)

13.11.1 General

The provisioning interface and procedures described herein do not describe a human-machine interface (HMI). This standard does not specify a specific HMI, but does describe how such tools can be designed for provisioning devices conforming to this standard. In plant operations, a user may enter provisioning data and accept or reject devices wishing to be provisioned, using a handheld device or an interface at some central location.

Provisioning scenarios in Clause 13 are examples of provisioning using methods described by this standard.

13.11.2 Examples of provisioning methods

13.11.2.1 General

Examples are discussed herein of how the described management objects and procedures can be used to provision a device. These examples use the following provisioning methods:

- provisioning over-the-air using pre-installed join keys;
- provisioning using out-of-band mechanisms;
- provisioning over-the-air using asymmetric-key (e.g., PKI) certificates;
- provisioning over-the-air using dual role advertisement routers; and
- provisioning backbone devices.

13.11.2.2 Provisioning over-the-air using pre-installed join keys

The steps for provisioning a device with pre-installed trust information may include:

a) The device arrives at the deployment site with pre-installed keys. The keys are symmetric join keys.

b) A WhiteList of device addresses and their corresponding symmetric keys is installed in the security manager of the target network. The mechanism by which these keys are installed in the security manager is beyond the scope of this standard. The WhiteList and corresponding symmetric keys may be securely emailed, sent on CDs, hand delivered and keyboard entered, or delivered using any other appropriate tool.

c) The target network may be using a subset of frequencies allowed and may be operating in the vicinity (i.e., in the interference range) of multiple distinct networks conforming to this standard. In this case, network-related information may be provisioned in the device. This information allows the device to respond to advertisements from target networks only, and also to listen for advertisements at the correct frequencies and at the correct time, thereby decreasing interference and decreasing join times. The network-related information may be provisioned using a PD via an out-of-band communication mechanism or via over-the-air mechanisms. When not provisioned with specific target network information, the device may try all channels and attempt to join all networks in its vicinity.

d) The device listens for advertisements from an advertising router in the target network. Once an advertisement is heard, the device sends a join request to the system/security manager of the target network. (The subnet joining process is described in 7.4.)

e) The system manager/security manager checks its WhiteList and then checks to see if the join key of the device matches the join key for the device provided to the security manager. If the join keys match, the security manager provides a master key to the device that is a shared secret key between the security manager and the device. In addition, T-keys and D-keys are provided, and a contract is established with the new device to complete the subnet joining process.

13.11.2.3 Provisioning using out-of-band mechanisms

The steps for provisioning a device using an OOB provisioning device might include:

a) A fresh device arrives at the user site. The device has default settings and no-pre-installed keys.

b) A PD (e.g., a handheld) obtains a list of symmetric keys generated by the security manager/system manager of the target network. The symmetric keys are time-bounded. The keys may be an array of keys or device-specific key/EUI64Address pairs at the discretion of the security manager/system manager. This information is stored in the PD.

c) The handheld device, loaded with the symmetric keys, is brought near the new device or connected to the new device and an OOB connection is made between the handheld

device and the DBP. The handheld device then uses the OOB communication interface to populate the attributes of the DPO in the new device. OOB communication may occur over infrared, physical connection, near field communication, or other means.

d) The device is now ready to join the target network and listens for advertisements from the target network. The device responds to the advertisements by sending a join request through the advertising router to the target system manager. The security manager checks its WhiteList, if applicable. The security manager also checks the validity of the join key and verifies that the key has not expired. If the key is valid, the security manager accepts the join request and provides the device with a master key that is a shared secret key between the security manager and the device. In addition, T-keys and D-keys are also provided and a contract is established with the new device to complete the subnet joining process.

13.11.2.4 Provisioning over-the-air using asymmetric key infrastructure certificates

The steps for provisioning a device that has pre-installed trust information might include:

a) The device arrives at the deployment site with an installed security module. The module contains a factory-signed certificate and a public/private key pair. A certificate authority (CA) has signed the issuer key of the factory.

b) A WhiteList of device addresses and a list of asymmetric keys of certificate authorities are installed in the security manager of the target network. The mechanism by which these keys are installed in the security manager is beyond the scope of this standard. The WhiteList may be securely emailed, sent on CDs, hand delivered and entered via keyboard, or delivered using any other appropriate tool.

c) The target network may be using a subset of the allowed channels, because it may be operating in the vicinity (i.e., within the interference range) of multiple networks. In such a case, network-related information may be provisioned to the device, enabling the device to respond to advertisements only from target networks and to listen for advertisements on the intended channels at appropriate times, thereby decreasing interference and increasing join rates. If the device is not provisioned with specific target network information, the device may try all channels and try to join all networks in its vicinity. The network-related information is provisioned using a provisioning device via an OOB communication mechanism or via over-the-air mechanisms.

d) The device now listens for the advertisements from the advertising router in the target network. Once an advertisement is heard, the device shares its certificates with the security manager. With the CA's public certificate, the security manager decodes the device certificate and checks that it is valid. The procedure also involves a challenge-response mechanism on the part of the system manager/security manager to confirm the identity of the joining device. The security manager checks its WhiteList to confirm that the device is intended to join the network. The confirmation step may involve a pop-up on a GUI of the security manager for manual confirmation by a user. Once confirmed, the security manager may issue T-keys for the device if the device wishes to join the network immediately. Alternatively, the security manager may issue symmetric join keys for the device to join the network at a later time. In either case, the issued keys are sent back to the device, encrypted with the public key of the device.

13.11.2.5 Provisioning over-the-air using dual role advertisement routers

The steps for provisioning a device over-the-air using a dual role advertisement router might include:

a) The device arrives with factory default settings at a user site. The user site requires very low levels of security.

b) Some of the advertisement routers at the user site have a dual role and function as provisioning devices. Using the open symmetric join key (K_join = K_open), the dual role advertisement router (i.e., the logical PD side of the dual role advertisement router) forms a mini-network with the new device and provides the new device with the network settings and join key for the target network. These settings, including the keys, are sent in the clear over-the-air. The dual role device may also inform the security manager of the target

network to update its white list by adding the device that has just been provisioned. The dual role provisioning device may be operational in a place where the user is fairly confident that transmission of join keys over-the-air poses little risk. (This step poses similar risk as that in binding garage door openers to remote controls.)

c) The DPO of the new device now has the trust information and network information to join the target network. It can use the advertisement routers (either the same dual role advertisement router that provisioned it, or some other advertisement router of the target network if the device was moved) and sends a join request to the system manager of the target network.

d) The system manager of the target network accepts the join request and provides a contract to the new device.

13.11.2.6 Provisioning backbone devices

The steps for provisioning backbone devices might include:

a) A fresh device arrives at the user site. The device has default settings and no pre-installed keys.

b) A PD (e.g., a handheld or a device connected via the backbone interface to the DBP) obtains a list of symmetric keys generated by the security manager/system manager of the target network. The symmetric keys are time-bounded. The keys may be an array of keys or device-specific key/EUI64Address pairs at the discretion of the security manager/system manager. This information is stored in the DPSO of the PD.

c) The PD loaded with the symmetric keys is brought near the new device or connected to the new device and an OOB connection (which can, in this case, be the backbone) is made between the handheld device and the DBP. The PD then uses the OOB communication interface (most probably the backbone interface) to populate the attributes of the DPO in the new device.

d) The device is now ready to join the target network. However, unlike a field device, a backbone device may not have a DL interface. For example, the device may be a gateway residing on the backbone. Alternatively, the backbone device may be the first advertising router connected to the network. For example, the device may be a backbone router with advertisement router functionality on the IEEE 802.15.4:2011 physical layer interface. However, the device needs to be provisioned over the backbone and not through the PhL, since in this case there are no advertising routers that can forward their join request to the system manager.

To talk to the system manager on the backbone without the help of an advertising router, the backbone router sends a join request to the system manager directly over the backbone; the backbone device can form the network header necessary to send this message. It can do so because it has been provisioned with the IPv6Address of the system manager (DPO.Target_System_Manager_Address). The remaining procedure at the system manager is the same as that in 13.11.2.3.

Annex A
(informative)

User layer/application profiles

A.1 Overview

Annex A describes what is meant by the terms “user layer” and “application profile”, and also describes how these terms relate to each other and to this standard.

A.2 User layer

The user layer is the term often applied to a non-existent eighth layer located atop the OSI seven-layer computer networking model. The intent of the user layer is to perform purpose-specific functions not related to network communications. With respect to industrial automation, the term “user layer” is sometimes applied to describe non-network communication-related hardware and/or software, such as a field sensor or a process control function block. It is possible that such a user layer has information that is to be communicated over another network that conforms to ISO/IEC 7498, the OSI Basic Reference Model.

The network communication to support the user layer function is initiated by the user layer employing the methods and protocols defined by the 7th layer of the OSI model, which is called the Application Layer (AL).

This standard is intended to support a variety of industrial automation industry functions that are not directly related to network communication. As such, it defines a general purpose communication stack compatible with the OSI computer networking model and includes the definition of AL standard services.

This standard also defines generic extensible standard objects, which may be used by industrial automation applications. This standard permits specialization of those standard objects, as well as definition of both new industry-specific standard objects and vendor-defined objects. The definition of industry-specific standard objects is outside the scope of this standard; that is left to each industry organization that promotes use of this standard for their industry. This standard does not limit the scope of user-layer functionality relative to any non-network-related communication need.

NOTE The ISA 100 Wireless Compliance Institute (WCI) is an example of such an organization for the process automation industries.

A.3 Application profile

An application profile defines application-specific properties to be implemented in a manner that fosters inter-operability among communicating entities. An application profile may also define implementation policies, and may suggest implementation guidelines. Any user layer within a device may implement one or more application profiles.

Some application-profile-specific properties may be mandatory for all instances of applications compliant with the particular application profile. Other application-profile-specific properties may be common practice properties that are construction options. All of these properties are represented as object attributes, so that communication of their values can occur through use of the basic application-layer services of this standard.

The scope of an application profile is often deliberately limited, in order to promote greater adoption and use of the particular application profile. An example of such a limited application profile is an application profile for temperature sensors.

In a loosely coupled system, the binding of devices that support application profiles with host system applications that employ those profiles usually is accomplished via the use of a device characterization file provided by the device vendor, the content of which often is based on a standard descriptive technology.

An example of a standard that may be used to describe profile content is IEC 61804-3, which may be used by industrial automation industry device vendors to create a file that may be used with appropriate host system companion tools, enabling the host to represent device functions, parameters (attributes) and their dependencies, graphical representations appropriate to data representation, as well as supported interactions with other devices.

A device may implement an application profile or set of profiles and may use the native AL methods and protocols of this standard to communicate over wireless networks conforming to this standard.

Because this standard is intended to support a variety of non-network communication-related industrial automation industry functions, this standard does not define or limit the definition or use of application profiles, languages or files that represent such devices, or tools used to represent such devices. The definition of industry-specific standard application profiles is therefore outside the scope of this standard. Instead it is delegated to those organizations that promote use of this standard in a particular automation industry.

NOTE ISO and IEC mechanisms exist for proposing such industry-specific application profiles.

Annex B
(normative)

Communication role profiles

B.1 Overview

B.1.1 General

A role profile is defined as the baseline capabilities, including any settings and configurations, that are required of a device to perform that role adequately. The roles are defined in 5.2.6.2, but are listed for reference here as system manager, security manager, backbone router, router, I/O, gateway, system time source, and provisioning device.

Annex B provides the role profile pro forma for compliance to this standard.

B.1.2 Purpose

The role profile will define those device capabilities, such as settings and configurations, necessary to fulfill each specific role defined in 5.2.6.2. The purpose for this is to ensure that devices complying with this standard, including Annex B, can be interworkable or interoperable, as appropriate, within the domain covered by the role profile.

B.1.3 System size

The capabilities required of a device to implement a role may be dependent upon the number of devices in the intended system. The minimum system size is defined in Clause 5, but there is no maximum system size. To allow the requirements of Annex B to serve a broad range of system sizes, those requirements dependent upon system size shall use a formula to specify the minimum capability. For the purposes of Annex B, the number of system devices is referred to as NSD.

B.1.4 Abbreviations and special symbols

Abbreviations and symbols used include:

- Notations for requirement status:
  - M: mandatory;
  - O: optional;
  - O.n: optional, but support of at least one of the group of options labeled O.n is required;
  - N/A: not applicable;
  - X: prohibited.
- Item: Conditional, status dependent upon the support marked for the item.

For example, a status of FD1:O.1 and FD2:O.1 indicates that the status is optional but at least one of the features described in FD1 and FD2 is required**.**

B.1.5 Role profiles

Table B.1 describes the protocol layers and media requirements for all role profiles. Should a device be declared to support more than one role, that device shall fulfill minimum capabilities for each role declared.

Table B.1 – Protocol layer device roles

Item number	Device role	Status					Reference	Support		
		Protocol layers			Medium					
		AL	TL	NL	Type A	Backbone		N/A	Yes	No
DR1	I/O	M	M	M	M	N/A	5.2.6.6			
DR2	Router	M	M	M	M	N/A	5.2.6.7			
DR3	Backbone router	M	M	M	M DR4: O DR7: O	M	5.2.6.9 5.2.6.9 5.2.6.9			
DR4	Gateway	M	M	M	DR2:O.1	DR3:O.1	5.2.6.10			
DR5	System time source	N/A	N/A	N/A	N/A	N/A	5.2.6.13			
DR6	Provisioning	M	M	M	M	N/A	5.2.6.8			
DR7	System manager	M	M	M	DR2:O.2	DR3:O.2	5.2.6.11			
DR8	Security manager	N/A	N/A	N/A	N/A	N/A	5.2.6.12			

B.2 System

The protocol of WISN supports the ability to upgrade devices over-the-air, as shown in Table B.2.

Table B.2 – Over-the-air upgrades

Item number	Role types affected	Reference	Status	Support		
				N/A	Yes	No
OTAR1	I/O		M			
OTAR2	Router		M			
OTAR3	Backbone router		N/A			
OTAR4	Gateway		N/A			
OTAR5	System manager		N/A			
OTAR6	Provisioning device		O			

B.3 System manager

The system manager allocates the ability for devices to communicate by generating, distributing, and maintaining contracts that define the resources necessary for that communication need. Since each device is required to store its contracts, the capacity of a device for contract storage is critical. While the necessary capacities of the I/O, router, and backbone router devices are dependent upon the number of application objects within those devices, the gateway and system manager are dependent upon the number of devices in the system, defined in Annex B as NSD. NSD does not include the system manager in its device count.

Contracts require communication sessions for communication, established by the security manager in conjunction with the system manager. Multiple contracts, communicating to the same endpoints, may share a single session. Minimum capacities described here assume that each session is matched with a single contract, recognizing that more contracts may be needed depending on the nature of the device's applications.

B.4 Security manager

The security manager establishes sessions between application processes. For example, when a device joins the network it needs a DMAP-SMAP session. The number of sessions that a device implementing a role shall be able to maintain is defined in Table B.3. The number of sessions supported by a system manager is dependent on NSD. The number of keys supported by a gateway is dependent on the number of Gateway-UAP connections that the gateway is designed to support, referred to as GUC in Table B.3.

An I/O device is presumed to require capacity to support the following sessions:

- A session between the device's DMAP and the SMAP, established when the device joins the network.
- A session between the device's UAP and a first device such as a gateway.
- A session between the device's DMAP and the first device, for reporting process alerts.
- A session between the device's UAP and a second device's UAP, such as for peer-to-peer communication.

Table B.3 – Session support profiles

Item number	Role types affected	Minimum number sessions supported	Comments	Status	Support		
					N/A	Yes	No
NCS1	I/O	4	DMAP-SMAP UAP-Gateway DMAP-Gateway UAP-Peer	M			
NCS2	Router	1	DMAP-SMAP	M			
NCS3	Backbone router	1	DMAP-SMAP	M			
NCS4	Gateway	(2 x GUC) + 1	DMAP-SMAP GUC x (Gateway-UAP) GUC x (Gateway-DMAP)	M			
NCS5	System manager	NSD	NSD x (SMAP-DMAP)	M			

The security manager assigns the security keys that are required for communication between devices. The number of keys that a device implementing a role shall be able to maintain is defined in Table B.4. The number of keys supported for a device depends on the number of sessions supported, with minimum capacities shown in Table B.3. In addition, each device needs capacity for a join key, a master key, and a D-key if a DL is included on the device. Key counts need to be doubled, because all keys except for the join key may be in the process of change-over.

Table B.4 – Baseline profiles

Item number	Role types affected	Minimum number keys supported	Comments	Reference	Status	Support		
						N/A	Yes	No
NKS1	I/O	1+((NCS1+2)×2)		7.2.2	M			
NKS2	Router	1+((NCS2+2) ×2)		7.2.2	M			
NKS3	Backbone router	1+((NCS3+2) ×2)		7.2.2	M			
NKS4	Gateway	1+((NCS4+1) ×2)	Add 2 if gateway has a DL	7.2.2	M			
NKS5	System manager	(NCS5+1) ×2	Add 2 if SM has a DL					
NKS7	Security manager	–N/A		7.2.2	N/A			

B.5 Physical layer

Since the PhL cites the specifications from IEEE 802.15.4, the role capabilities for the PhL are referenced in IEEE 802.15.4.

Table B.5 describes the physical layer roles.

Table B.5 – PhL roles

Item number	Item description		IEEE 802.15.4:2011 reference	Status	Support		
					N/A	Yes	No
PLR1	I/O	The device is a reduced function device	5.1	O.1			
		The device is a full function device	5.1	O.1			
PLR2	Router	The device is a full function device	5.1	M			
PLR3	Backbone router	The device is a full function device	5.1	M			
PLR4	Provisioning device	The device is a full function device	5.1	M			
O.1: at least one option shall be selected.							

B.6 Data-link layer

B.6.1 General

The DL affects four role profiles, as indicated in Table B.6.

Table B.6 – DL required for listed roles

Item number	Role types	Reference	Status	Support		
				N/A	Yes	No
DLR1	I/O	5.2.6.6	M			
DLR2	Router	5.2.6.7	M			
DLR3	Backbone router	5.2.6.9	M			
DLR4	Provisioning	5.2.6.8	M			

B.6.2 Role profiles

B.6.2.1 General

A DL role profile describes a set of minimum capabilities that shall be supported by every compliant device that implements the Type A field medium. For example, a device filling the router role shall support 8 neighbors. If a device meets all of the other requirements of a router, but supports only 4 neighbors, it is not compliant in its role as router. A device may exceed any of the requirements of its role, as long as all of the roles' minimum requirements are met.

The DL is configured through settings to the DL management object (DLMO) attributes, and the various roles are described as ranges of DLMO settings that a device can support.

B.6.2.2 DL management object attributes

A device's level of support for a capability can be expressed in relation to a set of DLMO attributes and elements of those attributes. Each attribute and/or element whose support varies by role is included.

If a number or range of numbers is listed, then a device filling this role shall support that number. If a single number is listed, it shall be interpreted as a minimum value unless indicated otherwise. For example, if a device shall support 3 neighbors, then it may support 4 neighbors, but is non-compliant if it supports only 2 neighbors. An I/O device may be capable of routing even if it isn't fully compliant with the router role; hence some capabilities related to routing are shown as optional (not prohibited) for an I/O device.

Table B.7 describes simple DLMO attributes with a single element. (The remaining tables in Annex B address DLMO attributes containing multiple elements.)

Table B.7 – Role profiles: General DLMO attributes

Attribute	Status			Comments	Support		
	I/O	Router	BBR		N/A	Yes	No
ActScanHostFract	O	M	M	A non-mains device will not necessarily have the energy to act as an active scanning host for an extended period of time. See Table B.8			
AdvJoinInfo AdvSuperframe	O	M	M	All routers and backbone routers can be configured to send advertisements			
TaiTime TaiAdjust	M	M	M	The DL is not necessarily the source of TaiTime for a particular device, and there are cases where a device's DL might not be involved in time propagation as a source or recipient. For example, a BBR might remain time synchronized through a backbone mechanism, and not be involved in DL time propagation			
ClockTimeout	M	M	M	A BBR may be configured as a clock recipient, but this is not intended as typical			

Table B.8 describes baseline role profiles for the dlmo.Device_Capability attribute. Those device elements not mentioned in Annex B shall be supported as described in Clause 9.

Table B.8 – Role profiles: dlmo.Device_Capability

Element	Status			Footnotes	Support		
	I/O	Router	BBR		N/A	Yes	No
QueueCapacity	0	10	20	a			
ClockStability	100	10	10	b			
DLRoles	0000 xxx1	0000 xx1x	0000 x11x	c			
AdvRate	0 (X)	6	6	d			
ListenRate	0 (X)	36	36	e			
TransmitRate	0 (X)	30	60	f			

a A system manager configures the DL queue only to the extent that the device is forwarding messages on behalf of other devices. The DL queue in a BBR is an internal device matter for graphs that originate or terminate in the device's DL.

b ClockStability values, as multiples of 1×10^{-6} are maximum allowed values over any continuous 30 s interval under industrial operating conditions. While low-cost I/O devices may have clocks with a short-term stability of only 100×10^{-6}, industrial I/O devices in general should have better stability. This standard was designed assuming that I/O devices have clocks with a short-term stability of 25×10^{-6} or better, and it is anticipated that most application profiles will be constrained accordingly.

c Bits indicate all of the DL roles that are supported by the device. Note that BBR is required to act as a router, such as for peer-to-peer messaging within a D-subnet.

d All devices serving router and backbone router roles shall have sufficient resources to transmit an advertisement every 10 seconds (6 DPDUs per minute), on average. See 9.1.17.

e All devices serving router and backbone roles shall have sufficient resources to operate their receivers for 36 s per hour (1 %), on average. A mains powered BBR will normally be capable of running its receiver continuously, but some BBR classes (such as wireless bridges) might be energy constrained.

f All devices serving router and backbone roles shall have sufficient resources to transmit the specified number of DSDUs per minute. See 9.1.17.

Table B.9 describes baseline role profiles for the dlmo.Ch attribute. Those device elements not mentioned in Annex B shall be supported as described in Clause 9.

Table B.9 – Role profiles: dlmo.Ch (channel-hopping)

Element	Status			Comments	Support		
	I/O	Router	BBR		N/A	Yes	No
Capacity (metadata)	10	10	10	Five default channel-hopping sequences, numbered 1..5, are defined by this standard. A device can be provisioned or configured with up to 5 additional channel-hopping sequences			
MaxRowID (metadata)	127	127	127	One octet			

Table B.10 describes baseline role profiles for the dlmo.TsTemplate attribute. Those device elements not mentioned in Annex B shall be supported as described in Clause 9.

Table B.10 – Role profiles: dlmo.TsTemplate

Element	Status			Comments	Support		
	I/O	Router	BBR		N/A	Yes	No
Capacity (metadata)	8	10	10	Three default timeslot templates, numbered 1..3, are defined by this standard. These are included in the capacity			
MaxRowID (metadata)	127	127	127	One octet			

Table B.11 describes baseline role profiles for the dlmo.Neighbor attribute. Those device elements not mentioned in Annex B shall be supported as described in Clause 9.

Table B.11 – Role profiles: dlmo.Neighbor

Element	Status			Comments	Support		
	I/O	Router	BBR		N/A	Yes	No
Capacity (metadata)	2	8	32	An I/O shall support at least two neighbors, so that it can maintain two active DL routes for reporting. A router adds additional capacity to support routing on behalf of neighbors			
MaxRowID (metadata)	2^{15}	2^{15}	2^{15}	6LoWPAN unicast address limited to 2^{15}			
GroupCode	O	M	M	GroupCode enables links to be used for multiple neighbors			
ExtendGraph	O	O	O	Automatic extension of graphs is required for all devices. Support for the ExtendGraph field is a construction option that provides a finer degree of control over graph extensions			

Table B.12 describes baseline role profiles for the dlmo.Diagnostic attribute. Those device elements not mentioned in Annex B shall be supported as described in Clause 9.

Table B.12 – Role profiles: dlmo.NeighborDiag

Element	Status			Comments	Support		
	I/O	Router	BBR		N/A	Yes	No
Capacity (metadata)	2 × 15 + 1 × 9	3 × 15 + 2 × 9	3 × 15 + 2 × 9	Diagnostic capacity (metadata) is measured in octets. Summary diagnostics, in Table 188, involve 15 octets of storage in the worst case. Actual storage and transmission may be more compact. Summary diagnostics are intended to be maintained on the “publication” side of a given link, to collect diagnostics from the direction where more traffic flows. Summary diagnostics include a baseline clock diagnostic (ClockSigma). More detailed clock diagnostics (Table 190) involve 9 octets of storage in the worst case. A summary clock diagnostic is provided along with the general diagnostic. Capacity is provided to collect these detailed clock diagnostics on an as-needed basis			
MaxRowID (metadata)	2^{15}	2^{15}	2^{15}	6LoWPAN unicast address limited to 2^{15}			

Table B.13 describes baseline role profiles for the dlmo.Superframe attribute. Those device elements not mentioned in Annex B shall be supported as described in Clause 9.

Table B.13 – Role profiles: dlmo.Superframe

Element	Status			Comments	Support		
	I/O	Router	BBR		N/A	Yes	No
Capacity (metadata)	3	5	10	Default superframes for discovery of provisioning device are included in this count			
MaxRowID (metadata)	127	127	127	One octet			
AlwaysHop	O	O	O	Support for this feature is a construction option			

Table B.14 describes baseline role profiles for the dlmo.Graph attribute. Those device elements not mentioned in Annex B shall be supported as described in Clause 9.

Table B.14 – Role profiles: dlmo.Graph

Element	Status			Comments	Support		
	I/O	Router	BBR		N/A	Yes	No
Capacity (metadata)	2	8	16				
MaxRowID (metadata)	127	127	127	One octet			

Table B.15 describes baseline role profiles for the dlmo.Link attribute. Those device elements not mentioned in Annex B shall be supported as described in Clause 9.

Table B.15 – Role profiles: dlmo.Link

Element	Status			Comments	Support		
	I/O	Router	BBR		N/A	Yes	No
Capacity (metadata)	9	15	30	Default links for discovery of provisioning device are included in this count			
MaxRowID (metadata)	127	127	127	One octet			
Discovery	0, 3	0, 1, 2, 3	0, 1, 2	Discovery refers to bits 3/2 in Table 182. A system manager may be configured to discover routing-capable neighbors through active or passive scanning for advertisements			
JoinResponse	O	M	M				
NeighborType=2	O	M	M	Support of neighbor groups is mandatory for routing devices			

Table B.16 describes baseline role profiles for the dlmo.Route attribute. Those device elements not mentioned in Annex B shall be supported as described in Clause 9.

Table B.16 – Role profiles: dlmo.Route

Element	Status			Comments	Support		
	I/O	Router	BBR		N/A	Yes	No
Capacity (metadata)	3	1	64	I/O device has capacity for routing to the system manager, a first device, and a second device. Router needs only a route to the system manager. BBR needs at least one route (outbound route lookup) for each device in its sphere of influence, even if those routes are identical to each other			
MaxRowID (metadata)	127	127	127	One octet			

Table B.17 describes baseline role profiles for the dlmo.Queue_Priority attribute. Those device elements not mentioned in Annex B shall be supported as described in Clause 9.

Table B.17 – Role profiles: dlmo.Queue_Priority

Element	Status			Comments	Support		
	I/O	Router	BBR		N/A	Yes	No
Capacity (metadata)	O	2	2				
MaxRowID (metadata)	127	127	127	One octet			

B.7 Network layer

Table B.18 describes role profiles for routing table sizes.

Table B.18 – Routing table size

Item number	Role types affected	Minimum number entries supported	Comments	Reference	Status	Support		
						N/A	Yes	No
RTS1	I/O	0			M			
RTS2	Router	0			M			
RTS3	Backbone router	15			M			

Table B.19 describes role profiles for address table sizes.

Table B.19 – Address table size

Item number	Role types affected	Minimum number entries supported	Comments	Reference	Status	Support		
						N/A	Yes	No
ATS1	I/O	4			M			
ATS2	Router	3			M			
ATS3	Backbone router	15			M			

B.8 Transport layer

Table B.20 describes role profiles for port support sizes.

Table B.20 – Port support size

Item number	Role types affected	Minimum number entries supported	Comments	Reference	Status	Support		
						N/A	Yes	No
PSS1	I/O	2			M			
PSS2	Router	1			M			
PSS3	Backbone router	1			M			

B.9 Application layer

Table B.21 describes the minimum number of APs per role.

Table B.21 – APs

Item number	Role types affected	Minimum number APs supported	Comments	Reference	Status	Support		
						N/A	Yes	No
UAPO1	I/O	2		Clause 6, 12.17	M			
UAP02	Router	1		Clause 6	M			
UAP03	Backbone router	1		Clause 6	M			
UAP04	Gateway	2		Clause 6, Annex U	M			
NOTE The maximum number of contained objects supported includes the UAPMO.								

B.10 Provisioning

Table B.22 provides the role profile devices implementing the I/O, router, gateway, or backbone router roles, all devices with a Type A field medium.

Table B.22 – Role profiles: I/O, routers, gateways, and backbone routers

Item number	Feature	Reference	Status	Range	Comments	Support		
						N/A	Yes	No
DBPR-1	Joining a provisioning network using K_global	13.6	M		See 7.2.2.2			
DBPR -2	Joining a provisioning network using K_open	13.6	O		Default value of K_join = K_open. Disabled once S is overwritten. Enabled again only if device reset to factory defaults			

B.11 Gateway (informative)

Table B.23 provides a notional role profile for a gateway.

Table B.23 – Role profile: Gateway

Item number	Feature	Reference	Status	Comments	Support		
					N/A	Yes	No
GWRP1	Native access	U.3.1.5	O.1	Allows native service access only			
GWRP2	Interworkable tunnel mechanism	U.3.1.5	O.1	Allows tunneled access only			

Table B.24 provides the notional role profile for a gateway native access.

Table B.24 – Role profile: Gateway native access

Item number	Feature	Reference	Status	Comments	Support		
					N/A	Yes	No
GWRP1.1	Min IFOs supported	U.3.1.5	1				
GWRP1.2	Buffered message behavior	U.3.4		Constant, static, dynamic, non-cacheable.			
GWRP1.3	Min devices	Table 373	NSD	NSD ≥ 5			
GWRP1.4	Min leases	Table 373	2 x NSD – 3	NSD ≥ 5			

Table B.25 provides the notional role profile for a gateway interworkable tunnel mechanism.

Table B.25 – Role profile: Gateway interworkable tunnel mechanism

Item number	Feature	Reference	Status	Comments	Support		
					N/A	Yes	No
GWRP2.1	Min TUNs supported	U.3.1.5	GD x AD + 1	GD ≥ 1 AD ≥ 5			
GWRP2.1.1	Supports a foreign protocol	U.3.1.5	Annex O				
GWRP2.1.2	2-part tunneling	U.3.1.5					
GWRP2.1.3	TUN objects with Array of Tunnel endpoints attributes with multiple address elements	U.3.1.5	1				
GWRP2.1.3.1	Number of elements in TUN with multiple address elements	U.3.1.5	A	A ≥ 5			
GWRP2.2	Min devices	Table 373	A	A ≥ 5			
GWRP2.3	Min leases	Table 373	2 x A	A ≥ 5			

Annex C
(informative)

Background information

C.1 Industrial needs

The wireless needs for industrial applications are significantly different than those required for residential, commercial, or military applications. These differences stem from the unique industrial ranking of priorities of characteristics such as device cost, system cost, lifecycle cost, reliability, maintainability, consistency, robustness, extensibility, security, coexistence, regulatory restrictions, interconnectability, and (within the relevant domains) interworkability or interoperability.

ISA 100 committee members collected and analyzed more than 500 use cases to define more completely the wireless communication needs of the industrial sector. The major conclusions of this effort were:

- Opportunity: Non-existent wireless sensing is an opportunity for end users, vendors, and emerging standards.
- Interworkable: Since multi-instrument-vendor facilities dominate the industrial environment, wireless standards should be of high value.
- Interoperable: Devices that target the same broad application domain (e.g., process control or asset management) should be interoperable with respect to basic functionality needed for cooperative action in that application domain.
- Integration: Multiple communication paths between devices are needed, especially to distributed control system (DCS) instruments.
- Applications: Applications such as monitoring/alerting are of greatest immediate interest since they constitute the largest potential use of wireless devices.
- Reliability and security: Critical factors for emerging standards and vendors.
- Power: Battery life expectations will vary due to application, environment, cost constraints, etc. Some devices will have mains power, while others will be powered by batteries or will scavenge energy from the environment.

C.2 Usage classes

C.2.1 General

While there are many techniques that may be used to categorize the communications needs of industrial applications, this standard uses classes based upon usage. Analysis of the patterns of intended use of inter-device industrial wireless communications resulted in a partitioning of such communications into six classes. These classes are summarized in Table C.1.

Table C.1 – Usage classes

Safety	Class 0: Emergency action	Always critical
Control	Class 1: Closed loop regulatory control	Often critical
	Class 2: Closed loop supervisory control	Usually non-critical
	Class 3: Open loop control	Human in the loop
Monitoring	Class 4: Alerting	Short-term operational consequence (e.g., event-based maintenance)
	Class 5: Logging and downloading/uploading	No immediate operational consequence (e.g., history collection, sequence-of-events, preventive maintenance)
NOTE Batch levels 3 and 4 could be class 2, class 1 or even class 0, depending on function. Batch levels are defined in IEC 61512-1, where L3 = unit and L4 = process cell.		

C.2.2 Class examples

- Class 0: Emergency action (always critical)

  Examples include:

  – safety interlock;
  – emergency shutdown;
  – automatic fire control.

- Class 1: Closed loop regulatory control (often critical)

  Examples include:

  – direct control of primary actuators (e.g., field device to host connection availability on demand of at least 99,99 %, with link outages > 500 ms intolerable, with demand rates of 0,2 Hz or greater);
  – high-frequency cascade loops.

- Class 2: Closed loop supervisory control (usually non-critical)

  Examples include:

  – low-frequency cascade loops;
  – multivariable controls;
  – optimizers.

- Class 3: Open loop control (human in the loop)

  Examples include:

  – operator manually initiates a flare and watches the flare;
  – guard remotely opens a security gate;
  – operator performs manual pump/valve adjustment.

- Class 4: Alerting – short-term operational consequence

  Examples include:

  – event-based maintenance;
  – marginal bearing temp results in technician sent to field;
  – battery low indicator for a device results in technician sent to change battery;
  – asset tracking.

- Class 5: Logging – data/messages with no immediate operational consequence

  Examples include:

  – history collection;
  – preventive maintenance rounds;

– sequence of events (SOE) uploading.

NOTE SOE uses lossless communication, such as file transfer, rather than timely communication such as used by control messaging.

C.2.3 Other uploading and downloading alarms (human or automated action)

Alarm examples include:

- Class 0: leak detector for radiation or fatally toxic gas, automated response (e.g., automated containment response).
- Class 1: high-impact process condition, automated response (e.g., automated shutdown of reaction).
- Class 2: automated response to process condition (e.g., automated flow diversion).
- Class 3: process condition with manually-initiated operational response (e.g., decide whether to divert flow to a parallel reactor).
- Class 4: equipment condition with short-time-scale maintenance response (e.g., send technician to field).
- Class 5: equipment condition with long-time-scale maintenance action (e.g., order spare parts).

C.3 The Open Systems Interconnection Basic Reference Model

C.3.1 Overview

This standard defines the protocol suite of the wireless network. A protocol suite is a particular software implementation of a networking protocol suite. In practical implementation, protocol suites are often divided into layers such as those defined by the Open Systems Interconnection Basic Reference Model ISO/IEC 7498-1. The format in this standard is based upon this reference model (see Figure C.1), implementing five of the basic reference model's seven layers.

NOTE It is useful to realize that this is a virtual model, which therefore imposes no actual requirements on implementations, or even specifications.

OSI layer	Function	IEC 62734 (and also ISA100.11a)
Application	Provides the user with network-capable application	• Uses object-oriented approach to encapsulate data and functionality in an extensible manner • Supports basic constructs of legacy automation protocols, extensible by industry groups and by vendors • Offers an open, interoperable application environment • Provides a common integration point to multiple host automation systems • Manages secured sessions between network devices
Presentation	Converts application data between network and local machine formats	
Session	Provides connection management services for applications	
Transport	Enables network-independent, transparent message transfer	Provides connectionless services based upon UDP with optional strong authentication, integrity, and confidentiality
Network	Provides end-to-end routing of packets; resolving network addresses	Provides network addressing, address translation, fragmentation (i.e., OSI segmentation) and reassembly, and network routing
Data link	Establishes data packet structure, framing, error detection, bus arbitration	Provides secure, robust, reliable links; time synchronization for time division multiple access, channel hopping and other uses
Physical	Provides mechanical/electrical connection; transmits raw bitstream	Uses 2,4 GHz band and IEEE 802.15.4 radios, thus eliminating in most countries the need for site licensing

Figure C.1 – OSI Basic Reference Model

The upper layer, application (AL), of the Basic Reference Model of this standard provides local functionality for one or more associated UAPs.

The four lower layers, transport (TL), network (NL), data-link (DL), and physical (PhL), are devoted to data communication. Each has the capability of multiplexing and demultiplexing, and of splitting and merging information flows from adjacent layers. In other words, the messaging relationships between an AL entity and a TL entity, or between a TL entity and an NL entity, or between an NL entity and a DL entity, or between a DL entity and a PhL entity, do not have to be one-to-one.

These lower layers also have the following abilities to:

- sequence service data units (SDUs) to maintain the order of original presentation;
- do one or more of the following
  - segment or reassemble SDUs into protocol data units (PDUs),
  - block or deblock SDUs into protocol data units (PDUs), and
  - concatenate or separate PDUs,

  so that they are sized more appropriately for the conveyance capabilities of the lower layer;
- split PDUs for conveyance over multiple lower layer routes, or to recombine such PDUs on receipt before forwarding on a higher-layer route; and
- acknowledge receipt of PDUs as a form of error control.

C.3.2 Application layer

The AL is the layer that interfaces directly to (and conceptually includes) UAPs, managing communications with other UAPs under the guidance of the local management UAP. A UAP may perform an individual function or any combination of functions. UAPs may be used, for example, to:

- handle input and/or output hardware;
- distribute communications to a set of co-resident UAPs within a device (proxy function);
- support tunneling of a non-native (e.g., control system legacy) protocol compatible with the network environment described in this standard; and/or
- perform a computational function.

The AL is usually composed of one or more UAPs that share common service elements.

The primary tasks of an AL entity are to provide:

- a place in the architecture of this standard for UAPs;
- the means by which UAPs manage communications with UAPs for other devices through the protocol suite, including:
  - identification of intended communications partners (e.g., by name, by address, by description, etc.),
  - agreement on security aspects (e.g., authentication, data integrity),
  - determination of acceptable quality of service (e.g., priority, time windows for control messaging, acceptability of out-of-order message delivery, acceptability of message delivery in partial increments, etc.),
  - agreement on responsibility for error recovery,
  - identification of abstract syntaxes, and
  - synchronization of cooperating UAPs;
- the means by which UAPs can inform the associated application entity of needed resource requirements, including those applicable to message buffering:
  - expected and maximum message sizes, and

  - maximum expected burstiness of message transmission and reception or how many messages can be sent or arrive within a short amount of time as compared to the average periodicity of messages; and
- any necessary communication functions that are not already performed by the lower layers.

C.3.3 Transport layer

The TL is the highest layer at which communicating applications are addressable. The primary tasks of a TL entity are:

- to provide addressing of UAPs via selection of a specific associated AL entity;
- to establish end-to-end messaging paths from one UAP to one or more other UAPs via their associated AL entities, where those processes are usually in separate devices;
- to convey and regulate the flow of messages between or among those UAPs; and
- to terminate those messaging paths when appropriate.

C.3.4 Network layer

The NL is the highest layer at which communicating devices are addressable. It is the lowest layer with more than local scope, which forwards messages between one entity group and others, or discards the messages. The primary tasks of an NL entity are:

- to provide network-wide addressing of devices;
- to relay messages (NPDUs) between entities (e.g., a router) via D-subnets, usually changing source and destination DL entity addresses associated with the message envelopes (DPDUs) in the process, or to discard the NPDUs; and
- to provide segmentation and reassembly of messages, as appropriate, to match the capabilities of the D-subnets on which messages are being forwarded.

NOTE The NL is the OSI layer where endpoint device addressing and routing occur. Lower layer relays are able to forward messages within a single addressing domain without message modification, but are unable to readdress messages or span addressing domains. Network-wide device addresses are IPv6Addresses.

C.3.5 Data-link layer

The DL is the lowest information-centric layer, which coordinates interacting PhL entities and provides basic low-level messaging among DL entities. The primary tasks of a DL entity are:

- to provide link-local addressing of peer-DL entities;
- to convey messages (DPDUs) from one DL entity to all others whose PhL entities are correspondents (e.g., to all PhL entities of the local link), or to discard the DPDUs;
- to manage use of the PhL;
- to provide low-level message addressing, message timing and message integrity checks;
- to provide low-level detection of and recovery from message loss (e.g., immediate acknowledgment; retry if no acknowledgment); and
- optionally, to relay DPDUs between DL entities (e.g., a bridge).

NOTE The DL is the OSI layer that manages and compensates for the specific characteristics of the selected physical communications technology. It provides only local addressing, and forwards messages within the local addressing domain without readdressing. It does not modify message addresses. DL16Addresses have only local scope, so it is possible that the same DL16Addresses are duplicated in other local links.

C.3.6 Physical layer

The PhL is the lowest layer of the OSI model and the only layer that deals with real-world physics. All other layers deal with abstract information, ultimately represented as bits; the PhL

is concerned with physical signals (sometimes referred to as baud or chips). The primary tasks of a PhLE are:

- to code bits, either singly or in multi-bit groups, into physical signals;
- to convey those signals from one physical location to another;
- to decode those signals into single-bit or multi-bit groups, possibly with error correction;
- to take direction from the associated DLE with respect to physical channel setup, physical receiver addressing and other aspects of the communications channel and coding;
- to convey to the locally-associated DLE information about the state of the PhLE, the channel and the last set of received signals; and
- optionally, to relay PhPDUs between PhLEs (e.g., a repeater).

Annex D
(normative)

Configuration defaults

D.1 General

Annex D summarizes the default settings for configuration.

D.2 System management

Table D.1 lists the system management configuration defaults.

Table D.1 – System management configuration defaults

Name	Initial default value	Reference
Confirmation_Timeout_Device_Diagnostics	10	Table 7
Alerts_Disable_Device_Diagnostics	0	Table 7
Confirmation_Timeout_Comm_Diagnostics	10	Table 7
Alerts_Disable_Comm_Diagnostics	0	Table 7
Confirmation_Timeout_Security	10	Table 7
Alerts_Disable_Security	0	Table 7
Confirmation_Timeout_Process	10	Table 7
Alerts_Disable_Process	0	Table 7
Comm_Diagnostics_Alarm_Recovery_AlertDescriptor	Default value: [FALSE, 3]	Table 7
Security_Alarm_Recovery_AlertDescriptor	Default value: [FALSE, 3]	Table 7
Device_Diagnostics_Alarm_Recovery_AlertDescriptor	Default value: [FALSE, 3]	Table 7
Process_Alarm_Recovery_AlertDescriptor	Default value: [FALSE, 3]	Table 7
DL_Alias_16_Bit	0	Table 10
Network_Address_128_Bit	0	Table 10
Device_Power_Status_Check_AlertDescriptor	Default value: [FALSE, 8]	Table 10
DMAP_State	1	Table 10
Join_Command	0	Table 10
Static_Revision_Level	0	Table 10
Restart_Count	0	Table 10
Uptime	0	Table 10
TAI_Time	0	Table 10
Comm_SW_Major_Version	0	Table 10
Comm_SW_Minor_Version	0	Table 10
System_Manager_128_Bit_Address	0	Table 10
System_Manager_EUI64	0	Table 10
System_Manager_DL_Alias_16_Bit	0	Table 10
Contract_Request_Timeout	30	Table 10
Max_ClientServer_Retries	3	Table 10
Max_Retry_Timeout_Interval	30	Table 10
DMAP_Objects_Count	1	Table 10

Name	Initial default value	Reference
Warm_Restart_Attempts_Timeout	60	Table 10
Current_UTC_Adjustment	35	Table 25
Next_UTC_Adjustment_Time	See Table 25	Table 25
Next_UTC_Adjustment	35	Table 25

D.3 Security

Table D.2 lists the security configuration defaults.

Table D.2 – Security configuration defaults

Name	Initial default value	Reference
Security_Level	1	Table 87
Protocol_Version	1	Table 92
DL_Security_Level	1	Table 92
Transport_Security_Level	1	Table 92
Join_Timeout	60 s	Table 92
MPDU_MIC_Failure_Limit	5	Table 92
MPDU_MIC_Failure_Time_Unit	60 s	Table 92
TPDU_MIC_Failure_Limit	5	Table 92
TPDU_MIC_Failure_Time_Unit	5	Table 92
DSMO_KEY_Failure_Limit	1	Table 92
DSMO_KEY_Failure_Time_Unit	1	Table 92
Security_MPDU_Fail_Rate_Exceeded_AlertDescriptor	[FALSE, 6]	Table 92
Security_TPDU_Fail_Rate_Exceeded_AlertDescriptor	[FALSE, 6]	Table 92
Security_Key_Update_Fail_Rate_Exceeded_AlertDescriptor	[FALSE, 6]	Table 92
pduMaxAge	510	Table 92
SoftLifeTime	50	Table 93
DSMO alert type 0 = Security_MPDU_Fail_Rate_Exceeded	0	Table 97
DSMO alert type 1 = Security_TPDU_Fail_Rate_Exceeded	0	Table 97
DSMO alert type 2 = Security_Key_Update_Fail_Rate_Exceeded	0	Table 97

D.4 Data-link layer

Table D.3 lists the DLE configuration defaults.

Table D.3 – DLE configuration defaults

Name	Initial default value	Reference
ActScanHostFract	0	Table 141
AdvJoinInfo	Null	Table 141
AdvSuperframe	0	Table 141
SubnetID	0	Table 141
SolicTemplate	Null	Table 141
AdvFilter	See 9.4.2.20	Table 141
SolicFilter	See 9.4.2.20	Table 141
TaiAdjust	Null	Table 141
MaxBackoffExp	5	Table 141
MaxDsduSize	96	Table 141
MaxLifetime	120 (30 s)	Table 141
NackBackoffDur	60 (15 s)	Table 141
LinkPriorityXmit	8	Table 141
LinkPriorityRcv	0	Table 141
EnergyDesign	See 9.4.2.22	Table 141
DeviceCapability	See 9.4.2.23	Table 141
IdleChannels	0	Table 141
ClockExpire	See 9.4.2.1	Table 141
ClockStale	45	Table 141
RadioSilence	600	Table 141
RadioSleep	0	Table 141
RadioTransmitPower	See 9.4.2.1	Table 141
CountryCode	0x3C00	Table 141
Candidates	Null	Table 141
DiscoveryAlert	60	Table 141
SmoothFactors	See Table 153	Table 141
QueuePriority	N=0	Table 141
Ch	See 9.4.3.2	Table 141
TsTemplate	See 9.4.3.3	Table 141
Neighbor	Empty	Table 141
Superframe	Empty	Table 141
Graph	Empty	Table 141
Link	Empty	Table 141
Route	Empty	Table 141
NeighborDiag	Empty	Table 141
ChannelDiag	See 9.4.2.27	Table 141
Transaction receiver template parameters	See Table 165	Table 165
Transaction initiator template parameters	See Table 166	Table 166
Transaction receiver template for scanning parameters	See Table 167	Table 167

D.5 Network layer

Table D.4 lists the NLE configuration defaults.

Table D.4 – NLE configuration defaults

Name	Initial default value	Reference
Enable_Default_Route	FALSE	Table 206
Max_NSDU_size	70	Table 206
Frag_Reassembly_Timeout	60	Table 206
Frag_Datagram_Tag	uniform random	Table 206
DroppedNPDUAlertDescriptor	[TRUE, 7]	Table 206
Source_Address*	0	Table 207
Destination_Address	0	Table 207
Contract_Priority	00	Table 207
Include_Contract_Flag	FALSE	Table 207
NWK_HopLimit	64	Table 208
Outgoing_Interface	0	Table 208

D.6 Transport layer

Table D.5 lists the TLE configuration defaults.

Table D.5 – TLE configuration defaults

Name	Initial default value	Reference
MaxNbOfPorts	15	Table 229
TPDUin	0	Table 229
TPDUinRejected	0	Table 229
TSDUout	0	Table 229
TSDUin	0	Table 229
TSDUinRejected	0	Table 229
TPDUout	0	Table 229
IllegalUseOfPortAlertDescriptor	[TRUE, 8] -- medium	Table 229
TPDUonUnregisteredPortAlertDescriptor	[TRUE, 4] -- low	Table 229
TPDUoutOfSecurityPoliciesAlertDescriptor	[TRUE, 2] -- journal	Table 229

D.7 Application layer

Table D.6 lists the ALE configuration defaults.

Table D.6 – ALE configuration defaults

Name	Initial default value	Reference
ObjectIdentifier	0	Table 240
UAP_ID	0=N/A	Table 240
UAP_TL_Port	0=N/A	Table 240
State	Active	Table 240
Command	0=None	Table 240
MaxRetries	3	Table 240
Number of unscheduled communication correspondents	0=N/A	Table 240
Number of objects in the UAP including this UAPMO	1	Table 240
Static_Revision_Level	0	Table 240
Categories	0	Table 243
Errors	0	Table 243
State	0=Idle	Table 246
MaxBlockSize	1..(MaxNPDUsize + Max TL header size – max(sizeof (additional coding of AL UploadData service request), additional coding of sizeof(AL DownloadData service response))	Table 246
MaxDownloadSize	0	Table 246
MaxUploadSize	0	Table 246
DownloadPrepTime	0	Table 246
DownloadActivationTime	0	Table 246
UploadPrepTime	0	Table 246
UploadProcessingTime	0	Table 246
DownloadProcessingTime	0	Table 246
CutoverTime	0	Table 246
LastBlockDownloaded	0	Table 246
LastBlockUploaded	0	Table 246
ErrorCode	0 =(noError)	Table 246
Revision	0	Table 256
CommunicationEndpoint	The configured connection endpoint valid element indicates not configured (i.e., endpoint is not valid)	Table 256
MaximumItemsPublishable	Local matter	Table 256
NumberItemsPublishing	0	Table 256
Array of ObjectAttributeIndexAndSize	Element size is 0	Table 256
Concentrator ContentRevision	0	Table 258
CommunicationEndpoint	The configured connection endpoint valid element indicates not configured (i.e., endpoint is not valid)	Table 258
MaximumItemsSubscribing	Local matter	Table 258
NumItemsSubscribing	0	Table 258

Name	Initial default value	Reference
Array of ObjectAttributeIndexAndSize	Element size is 0	Table 258
Protocol	Local matter (protocol-specific)	Table 260
Status (Configuration status)	0	Table 260
Period (Data publication period)	0	Table 260
Max_Peer_Tunnels	0	Table 260
Num_Peer_Tunnels	0	Table 260
ObjectIdentifier	7	Table 283
MalformedAPDUsAdvise	FALSE	Table 283
TimeIntervalForCountingMalformedAPDUs	0	Table 283
MalformedAPDUsThreshold	0	Table 283
MalformedAPDUAlertDescriptor	[TRUE, 7]	Table 283
PV	NaN	Table 287
Mode	OOS	Table 287
Scale	Engineering units values for 0 % and for 100 % BOTH indicate 0	Table 287
OP	NaN	Table 290
Mode	OOS	Table 290
Readback	NaN	Table 290
Scale	Engineering units values for 0 % and for 100 % BOTH indicate 0	Table 290
PV_B	0	Table 293
Mode	Read only for actual mode; target mode, permitted mode, and normal mode all have read/write access	Table 293
OP_B	0	Table 296
Mode	Read only for actual mode; target mode, permitted mode, and normal mode all have read/write access	Table 296
Readback_B	0	Table 296
Target	OOS	Table 302
Actual	OOS	Table 302
Permitted	OOS	Table 302
Normal	OOS	Table 302
Engineering units at 100 %	0	Table 304
Engineering units at 0 %	0	Table 304
Decimal point location	0	Table 304

D.8 Provisioning

Table D.7 lists the provisioning configuration defaults.

Table D.7 – Provisioning configuration defaults

Name	Initial default value	Reference
Default_NWK_ID	0x0001	Table 368
Default_SYM_Join_Key	K_global	Table 368
Open_SYM_Join_Key	K_open	Table 368
Default_Channel_List	0x7FFF	Table 368
Join_Method_Capability	00	Table 368
Allow_Provisioning	TRUE (1)	Table 368
Allow_Over_The_Air_Provisioning	TRUE (1)	Table 368
Allow_OOB_Provisioning	TRUE (1)	Table 368
Allow_Reset_to_Factory_defaults	TRUE (1)	Table 368
Allow_Default_Join	TRUE (1)	Table 368
Target_NWK_ID	0	Table 368
Target_NWK_BitMask	0xFFFF	Table 368
Target_Join_Method	1 (asymmetric key)	Table 368
Number of PKI_Certificates	1	Table 368
Current_UTC_Adjustment	35	Table 368
White_List	[ ]	Table 371
Symmetric_Key_List	{K_global}	Table 371
Symmetric_Key_Expiry_Times	{0xFFFF FFFF}	Table 371
Target_NWK_ID	0	Table 371
Target_Join_Method	1 (asymmetric key)	Table 371
Target_Join_Time	0	Table 371
Allow_Provisioning	TRUE (1)	Table 371
Allow_Default_Join	TRUE (1)	Table 371
Device_Specific_Provisioning_Flag	disabled (0)	Table 371
DPSO_Alerts_AlertDescriptor	[FALSE, 6]	Table 371
Current_UTC_Adjustment	35	Table 371
Device_EUI	0x0000 0000 0000 0001	Table 372
Device_Symmetric_Key	K_global	Table 372
Device_Symmetric_Key_Expiry_Time	{0xFFFF FFFF}	Table 372
Target_NWK_ID	0	Table 372
Target_Join_Method	1 (asymmetric key)	Table 372
Allow_Provisioning	TRUE	Table 372
Allow_Default_Join	TRUE	Table 372

D.9 Gateway (informative)

Table D.8 lists the gateway configuration defaults.

Table D.8 – Gateway configuration defaults

Name	Initial default value	Reference
Max_Devices	0	Table U.41

Annex E
(informative)

Use of backbone networks

E.1 General

Use of a backbone network can be advantageous to the system designer, since it takes the message off of the Type A field medium, allowing additional bandwidth and higher QoS for other messages.

E.2 Recommended characteristics

Although the backbone itself is not specified within this standard, it is assumed and recommended that the backbone will have the following characteristics:

- Throughput equal to or greater than the throughput of the Type A field medium (≥ 250 kbit/s).
- Capability of supporting two-way unsolicited message traffic.
- Quality of service of a sufficient level such that time synchronization can be maintained across the network. This may place specific time synchronization methods on the backbone.
- High reliability. Operation should not burden the network with frequent lost messages and retries.
- Security sufficient not to present a security threat to the end users application or the network.
- Capability of either encapsulating (tunneling) protocol TPDUs or TSDUs defined by this standard or translating them in such a way that they may traverse the backbone without being modified when emerging at the backbone devices. In general, the backbone shall be able to take a standard-compliant TSDU from the point of ingress and deliver it across the backbone to the point of egress unmodified.
- Capability of preserving the end-to-end application security mechanisms.
- Support for multipoint networking between devices.

It is recognized that many standard fieldbuses may not have these characteristics and therefore may not be suitable for use as a backbone network. In many cases, a backbone network will be an IP network such as ISO/IEC 8802-3 (IEEE 802.3) or ISO/IEC 8802-11 (IEEE 802.11), but there is no requirement for this. There are many other alternatives in the marketplace that exist and are well-suited for the purposes of a backbone network. These might include simple point-to-point or point-to-multipoint wireless networks.

E.3 Internet protocol backbones

E.3.1 Methods of IPv6 protocol data unit transmission

In many cases, an available backbone will use an Internet protocol (IP) NL. In this case there are many different ways to transport the wireless industrial sensor network (WISN) TPDUs using standardized protocol behavior:

- Encapsulate wireless industrial sensor network transport protocol data units within IPv4 NPDUs.

  The mechanism used to encapsulate WISN TPDUs within IPv4 NPDUs is formally known as IPv6 over IPv4 or 6over4 and is sometimes called virtual Ethernet. This

method is documented in IETF RFC 2529. A backbone router – IETF RFC 2529 refers to them as IPv6 hosts – located on a physical link that has no directly connected IPv6 router may become a fully functional IPv6 host by using an IPv4 multicast domain as its virtual local link. Backbone routers connected using this method do not require IPv4-compatible addresses or configured tunnels.

- Tunnel wireless industrial sensor network transport protocol data units over IPv4 network.

  Following IETF RFC 4213, this method encapsulates IPv6 protocol data units (PDUs) within IPv4 headers to carry them over IPv4 routing infrastructures. Two types of tunneling are possible, configured and automatic. In configured tunneling, the IPv4 tunnel endpoint address is determined by configuration information on the encapsulating node. In automatic tunneling, the IPv4 tunnel endpoint address is determined from the IPv4 address embedded in the IPv4-compatible destination address of the IPv6 PDU.

- Encapsulate wireless industrial sensor network transport protocol data units within raw Ethernet DPDUs.

  This method specifies the NPDU format for transmission of IPv6 PDUs following IETF RFC 2464. Furthermore, this method dictates the formation of link-local IPv6Addresses and statelessly-autoconfigured addresses on Ethernet networks. Finally, this approach specifies the content of the source/target link-layer addresses used in router solicitation, router advertisement, neighbor solicitation, neighbor advertisement, and redirect messages when those messages are transmitted on an Ethernet network.

- Use native IPv6 backbone without any encapsulation or tunneling.

  If the backbone uses an IPv6 NL, neither encapsulation nor tunneling is necessary, since the backbone native mode is to transport IPv6 PDUs.

E.3.2 Backbone router peer device discovery

For the backbone router (BBR) to function properly and to connect WISN devices on the backbone, it needs to know the backbone addresses of the other BBRs or peers in the backbone network. Within each BBR, the addressing information of its peers should be stored in a backbone router peer table (BRPT). There are two basic methods of generating the BRPT, configuration and discovery.

NOTE The BRPT and the mechanism for discovering peers are beyond the scope of this standard.

Configuration occurs when the addresses of peer BBRs are inserted into the BRPT by the system manager or an operator. The advantages of this method are that it is straightforward and prevents the BBR from accessing inappropriate devices on the backbone.

Discovery occurs when the BBR automatically searches the backbone for peer devices. There are multiple discovery techniques, such as those mentioned in IETF RFC 2529 and others. The advantages of this method are that it is automatic, requires no operator involvement, and can be easily and often updated.

E.3.3 Security

E.3.3.1 Security of transport protocol data units

The security mechanisms of the backbone are beyond the scope of this standard. Typical IP security methods include IPSec, SSL, and others. In addition to any security mechanisms on the backbone, the WISN TL security mechanism protects the TPDU within the backbone.

E.3.3.2 Security of the backbone

There is a perception by some that allowing a WISN to access an IP backbone could degrade the security of the backbone. This concern may be mitigated by restricting the BBR access to

only peer BBRs via an access control list or by the use of firewalls set up to restrict access properly to specific devices.

Annex F
(normative)

Basic security concepts – Notation and representation

F.1 Strings and string operations

A string is a sequence of symbols over a specific set (e.g., the binary alphabet (0, 1) or the set of all octets).

The size of a string is the number of symbols it contains (over the same alphabet).

The right-concatenation of two strings x and y of size m and n respectively (notation $x \,||\, y$) is the string z of size $m + n$ that coincides with x on its leftmost m symbols and with y on its rightmost n symbols.

An octet is a symbol string of size 8. In this context, all octets are strings over the binary alphabet.

F.2 Integers, octets, and their representation

Throughout this standard, the representation of integers as octet strings and of octet strings as binary strings shall be fixed.

All integers shall be represented as octet strings in most-significant-octet-first order. This representation conforms to the convention in ANSI X9.63:2011, 4.3.

All octets shall be represented as binary strings in most-significant-bit-first order.

F.3 Entities

Throughout this standard, each entity shall be a DEV and shall be uniquely identified by its EUI64Address. The parameter entityIdSize shall have the value 64.

Annex G
(informative)

Using certificate chains for over-the-air provisioning

This standard uses the implicit certificate called the PublicReconstrKey (see Annex H for details) for the asymmetric key-based cryptography. Given the identity of a device A (ID_A) and the implicit certificate γ_A of the device, the public key of the device A can be computed using the following equation:

$$Q_A = Hash(\gamma_A||ID_A)\gamma_A+Q_{CA}$$

where Q_{CA} is the public key of the certificate authority (CA).

With this background, the following steps outline the process for OTA provisioning using asymmetric-key cryptography as outlined in Figure 137.

1) The CA publishes Q_{CA}, its public key, on the web.
2) The device manufacturer (DM) gets a certificate from the CA:

   C_DM = PublicReconstrKey(DM) || Subject(DM) || Issuer(CA) || Text

   where:

   - Subject = ID of the DM
   - Issuer = ID of the CA
   - PublicReconstrKey_DM = γ_DM is used to calculate the public key of the DM using the equation:

     $Q_{DM} = HASH(\gamma\_DM||Subject)\gamma\_DM + Q_{CA}$
3) The individual device gets a certificate from the DM:

   C_DEV = PublicReconstrKey(DEV) || Subject(DEV) || Issuer(DM) || Text

   where:

   - Subject = ID of the device
   - Issuer =ID of the DM
   - PublicReconstrKey_DEV = γ_DEV is used to calculate the public key of the device using the equation:

     $Q_{DEV} = HASH(\gamma\_DEV||Subject)\gamma\_DEV + Q_{DM}$
   - C_DEV and C_DM are populated in the DBP by the DM.
4) The DBP joins the PD in a provisioning network. The PD has QCA.
5) The DBP sends a random number, C_DEV, and C_DM to the PD. The PD calculates Q_DEV as explained in steps 2) and 3).
6) A challenge/response mechanism is used to authenticate the device, and the security manager should validate the manufacturer's implicit certificate at this point.
7) If the challenge/response is passed, the PD sends K_join encrypted with Q_DEV.

Annex H
(normative)

Security building blocks

H.1 Symmetric key cryptographic building blocks

H.1.1 Overview

The following symmetric key cryptographic primitives and data elements are defined for use with all security processing operations specified in this standard.

H.1.2 Symmetric key domain parameters

The symmetric key shall have key size keylen=128 (in bits).

H.1.3 Block cipher

The block cipher used in this standard shall be AES-128, as specified in ISO/IEC 18033-3. This block cipher shall be used with symmetric keys as specified in H.1.2. In this case the key size is equal to the block size of the block cipher, 128 bits.

H.1.4 Mode of operation

The block-cipher mode of operation used in this standard shall be the CCM* mode of operation, as specified in IEEE 802.15.4:2011, B.3.2.

H.1.5 Cryptographic hash function

The cryptographic hash function used in this standard shall be the block cipher-based cryptographic hash function specified in Clause H.9, with the following instantiations:

- each entity shall use the block-cipher E as specified in H.1.3;
- all integers and octets shall be represented as specified in Clause F.2.

The Matyas-Meyer-Oseas hash function (see Clause H.9) has a message digest size hashlen that is equal to the block size, in bits, of the established block cipher.

H.1.6 Keyed hash function for message authentication

The keyed hash message authentication code (HMAC) used in this standard shall be HMAC, as specified in the FIPS 198, with the following instantiations:

- each entity shall use the cryptographic hash H function as specified in H.1.5;
- the block size B shall have the integer value B=keylen/8, where keylen is as specified in H.1.2 (i.e., B is equal to the size of the symmetric key, in octets, that is used by the keyed hash function);
- the output size HMAClen of the HMAC function shall have the same integer value as the message digest parameter hashlen, as specified in H.1.5.

H.1.7 Specialized keyed hash function for message authentication

The specialized[12] keyed hash message authentication code used in this standard shall be the keyed hash message authentication code, as specified in H.1.6.

H.1.8 Challenge domain parameters

The challenge domain parameters used in this standard shall be as specified in H.6.2, with the instantiation (minchallengelen, maxchallengelen)=(keylen, keylen), where keylen is as specified in H.1.2.

All challenges shall be validated using the challenge validation primitive as specified in Clause H.7.

H.2 Asymmetric-key cryptographic building blocks

H.2.1 General

The following asymmetric-key cryptographic primitives and data elements are defined for use with all security processing operations specified in this standard.

NOTE See also ISO/IEC 18033-2 for more information on asymmetric cryptography.

H.2.2 Elliptic curve domain parameters

The elliptic curve domain parameters used in this specification shall be those for the curve ansit283k1 as specified in ANSI X9.63:2011, Appendix J4.5, example 1.

All elliptic curve points shall be validated using the public key validation primitive as specified in ANSI X9.63:2011, 5.2.2.

H.2.3 Elliptic curve point representation

All elliptic curve points shall be represented in polynomial notation as specified in ANSI X9.63:2011, 4.1.2.1. All elliptic curve points shall be transmitted in compressed form, as specified in ANSI X9.63:2011, 4.2.2.

H.2.4 Elliptic curve public-key pair

An elliptic curve-key pair consists of an integer q and a point Q on the curve determined by multiplying the generating point G of the curve by this integer (i.e., Q=qG) as specified in ANSI X9.63:2011. Here, Q is called the public key, whereas q is called the private key; the pair (q, Q) is called the public-key pair. Each private key shall be represented as specified in ANSI X9.63:2011, 4.3.1. Each public key shall be on the curve as specified in H.2.2 and shall be represented as specified in H.2.3.

H.3 Keying information

H.3.1 General

The following specifies the format of asymmetric-key keying information used in this standard.

---

12 This refers to a MAC scheme where the MAC function has the additional property that it is also pre-image and collision resistant for parties knowing the key (see also remark 9.8 of Menezes et al.). Such MAC functions allow key derivation in contexts where unilateral key control is undesirable.

H.3.2 Elliptic curve cryptography implicit certificates

Implicit certificates IC_U shall be specified as

IC_U = PublicKeyReconstrData || Subject || Issuer || Usage_Serial || KeyValidityInfo || Text

where:

- The parameter PublicKeyReconstrData shall be the public-key reconstruction data BEU as specified in the implicit certificate generation scheme (see H.5.1).
- The parameter Subject shall be the entity U that is bound to the public key reconstruction data BEU during execution of the implicit certificate generation scheme, i.e., the entity that purportedly owns the private key corresponding to the public key that can be reconstructed from PublicReconstrKey.
- The parameter Issuer shall be the entity of the certificate authority (CA) that creates the implicit certificate during the execution of the implicit certificate generation scheme.
- The parameter Usage_Serial is defined in Table 68.
- The parameter KeyValidityInfo shall indicate the validity period of the keying material as indicated by the parameters ValidNotBefore and ValidNotAfter, which indicate the beginning and the end of the validity period, respectively. The KeyValidityInfo shall be formatted as

  KeyValidityInfo = ValidNotBefore || ValidNotAfter

  where ValidNotBefore and ValidNotAfter shall be represented as specified in 12.22.4.2.
- The parameter Text shall be the representation of additional information, as specified in H.3.4.
- The string I_U as specified in the implicit certificate generation scheme (see H.5.1) shall be the octet string consisting of the octet strings Subject, Issuer, and Text, as follows:

  I_U = Subject || Issuer || Text

H.3.3 Elliptic curve cryptography manual certificates

Manual certificates MC_U shall be specified as MC_U = PublicKey || Subject || Issuer || Text, where:

- The parameter PublicKey shall be the octet representation of the public key W_U as specified in the manual certificate generation transformation.
- The parameter Subject shall be the entity U of the purported owner of the private key corresponding to the public key represented by PublicKey.
- The parameter Issuer shall be the entity of the CA that creates the manual certificate during the execution of the manual certificate generation transformation (the so-called certificate authority).
- The parameter Usage_Serial is defined in Table 68.
- The parameter KeyValidityInfo shall indicate the validity period of the keying material as indicated by the parameters ValidNotBefore and ValidNotAfter, which indicate the beginning and the end of the validity period, respectively. The KeyValidityInfo shall be formatted as

  KeyValidityInfo = ValidNotBefore || ValidNotAfter

  where ValidNotBefore and ValidNotAfter shall be represented as specified in.12.22.4.2.
- The parameter Text shall be the representation of additional information, as specified in H.3.4.

- The string I_U as specified in the manual certificate scheme (see Clause H.10) shall be the octet string consisting of the octet strings Subject, Issuer, and Text, as follows:

  I_U = Subject || Issuer || Text

NOTE A manual certificate is not a real digital certificate, since the binding between the PublicKey and the Subject is established and verified by non-cryptographic means.

H.3.4 Additional information

Additional information Text shall be specified as follows:

Text = Reserved

where the parameter Reserved allows for future extensions of the additional information and shall be set to the all-zero bit string for this version of the standard.

H.4 Key agreement schemes

H.4.1 Symmetric-key key agreement scheme

The symmetric-key key agreement scheme used in this standard shall be the full symmetric-key with key confirmation scheme as specified with the following instantiations:

- Each entity shall be identified as specified in Clause F.3.
- Each entity shall use the HMAC-scheme as specified in H.1.5.
- Each entity shall use the cryptographic hash function as specified in H.1.5.
- The parameter keydatalen shall have the same integer value as the key size parameter keylen as specified in H.1.2.
- Each entity shall use the challenge domain parameters as specified in H.1.8.
- All octets shall be represented as specified in Clause F.2.

H.4.2 Asymmetric-key key agreement scheme

The asymmetric-key key agreement scheme used in this standard shall be the full MQV with key confirmation scheme as specified in ANSI X9.63:2011, 6.11, with the following instantiations:

- Each entity shall be identified as specified in Clause F.3.
- Each entity shall use the HMAC-scheme as specified in H.1.5.
- Each entity shall use the cryptographic hash function as specified in H.1.5.
- The parameter keydatalen shall have the same integer value as the key size parameter keylen as specified in H.1.2.
- The parameter SharedData shall be the empty string; parameter shareddatalen shall have the integer value 0.
- Each entity shall use the elliptic curve domain parameters as specified in H.2.2.
- All elliptic curve points shall be represented as specified in H.2.3.
- All octets shall be represented as specified in Clause F.2.

H.5 Keying information schemes

H.5.1 Implicit certificate scheme

The implicit certificate scheme used in this standard shall be the ECQV implicit certificate scheme as specified in SEC 4, with the following instantiations:

- Each entity shall be identified as specified in Clause F.3.
- Each entity shall use the cryptographic hash function as specified in H.1.5.
- Each entity shall use the elliptic curve domain parameters as specified in H.2.2.
- All elliptic curve points shall be represented as specified in H.2.3.
- All implicit certificates shall be represented as specified in H.3.2.
- The implicit certificate infrastructure shall be one of the schemes as specified in H.3.2.
- All octets shall be represented as specified in Clause F.2.

H.5.2 Manual certificate scheme

The manual certificate scheme used in this standard shall be the manual certificate scheme as specified in Clause H.10, with the following instantiations:

- Each entity shall be identified as specified in Clause F.3.
- Each entity shall use the elliptic curve domain parameters as specified in H.2.2.
- All elliptic curve points shall be represented as specified in H.2.3.
- All manual certificates shall be represented as specified in H.3.2.
- The manual certificate infrastructure shall be one of the schemes as specified in H.3.2.
- All octets shall be represented as specified in Clause F.2.

H.6 Challenge domain parameter generation and validation

H.6.1 Overview

Challenge domain parameters impose constraints on the size(s) of bit challenges that a scheme expects. As such, this determines a bound on the entropy of challenges and, thereby, on the security of the cryptographic schemes in which these challenges are used. In most schemes, the challenge domain parameters will be such that only challenges of a fixed size will be accepted (e.g., 128-bit challenges). However, one may define the challenge domain parameters such that challenges of varying size might be accepted. The latter is useful in contexts wherein entities that wish to engage in cryptographic schemes might have a defective or low-quality random bit generator. Allowing both entities that engage in a scheme to contribute sufficiently long inputs enables each of these to contribute sufficient entropy to the scheme at hand.

In this standard, challenge domain parameters will be shared by a number of entities using a scheme of this standard. The challenge domain parameters may be public; the security of the system does not rely on these parameters being secret.

H.6.2 Challenge domain parameter generation

Challenge domain parameters shall be generated using the following routine:

- Input: This routine does not take any input.
- Actions: The following actions are taken:
  - Choose two nonnegative integers minchallengelen and maxchallengelen, such that minchallengelen ≤ maxchallengelen.
- Output: Challenge domain parameters D = (minchallengelen, maxchallengelen).

H.6.3 Challenge domain parameter verification

Challenge domain parameters shall be verified using the following routine:

- Input: Purported set of challenge domain parameters D=(minchallengelen, maxchallengelen).
- Actions: The following checks are made:
  - Check that minchallengelen and maxchallengelen are nonnegative integers.
  - Check that minchallengelen ≤ maxchallengelen.
- Output: If any of the above verifications has failed, then output invalid and reject the challenge domain parameters. Otherwise, output valid and accept the challenge domain parameters.

H.7 Challenge validation primitive

Challenge validation refers to the process of checking the size properties of a challenge. It is used to check whether a challenge to be used by a scheme in this standard has sufficient size (e.g., messages that are too short are discarded, due to insufficient entropy).

The challenge validation primitive is used in Clause H.7 and uses the following routine:

- Input: The input of the validation transformation is a valid set of challenge domain parameters D = (minchallengelen, maxchallengelen), together with the bit string Challenge.
- Actions: The following actions are taken:
  - Compute the bit-length challengelen of the bit string Challenge.
  - Verify that challengelen ∈ [minchallengelen, maxchallengelen]. (That is, verify that the challenge has an appropriate size.)
- Output: If the above verification fails, then output invalid and reject the challenge. Otherwise, output valid and accept the challenge.

H.8 Secret key generation (SKG) primitive

The SKG primitive derives a shared secret value from a challenge owned by an entity U_1 and a challenge owned by an entity U_2 when all the challenges share the same challenge domain parameters. If the two entities both correctly execute this primitive with corresponding challenges as inputs, the same shared secret value will be produced.

The shared secret value shall be calculated as follows:

- Prerequisites: The following are the prerequisites for the use of the SKG primitive:
  - Each entity shall be bound to a unique identifier (e.g., distinguished names). All identifiers shall be bit strings of the same size, entityIdSize. Entity U_1's identifier will be denoted by the bit string U_1. Entity U_2's identifier will be denoted by the bit string U_2.
  - A specialized[13] MAC scheme shall have been chosen, with tagging transformation as specified in ANSI X9.63:2011, 5.7.1. The size in bits of the keys used by the specialized MAC scheme is denoted by macKeySize.
- Input: The SKG primitive takes as input:
  - A bit string MACKey of size macKeySize bits to be used as the key of the established specialized MAC scheme.
  - A bit string QEU_1 owned by U_1.
  - A bit string QEU_2 owned by U_2.

[13] This refers to a MAC scheme wherein the MAC function has the additional property that it is also pre-image- and collision-resistant for parties knowing the key (see also remark 9.8 of Menezes et al.). Such MAC functions allow key derivation in contexts where unilateral key control is undesirable.

- Actions: The following actions are taken:
  - Form the bit string consisting of U_1's identifier, U_2's identifier, the bit string QEU_1 corresponding to U_1's challenge, and the bit string QEU_2 corresponding to U_2's challenge:
  - MacData = $U_1 \,||\, U_2 \,||\, QEU_1 \,||\, QEU_2$.
  - Calculate the tag MacTag for MacData under the key MacKey using the tagging transformation of the established specialized MAC scheme:
  - MacTag = MAC_{MacKey}(MacData).
  - If the tagging transformation outputs invalid, output invalid and stop.
  - Set Z=MacTag.
- Output: The bit string Z as the shared secret value.

H.9 Block-cipher-based cryptographic hash function

The Matyas-Meyer-Oseas hash function is a cryptographic hash function based on block-ciphers. This hash function is defined for block-ciphers with a key size that is equal to the block size, such as AES-128, and with a particular choice for the fixed initialization vector IV (which here is defined to be IV=0).

NOTE For a more general definition of the Matyas-Meyer-Oseas hash function, see *Handbook of applied cryptography*:1996, 9.4.1, listed in the Bibliography.

The hash function is defined as follows:

- Prerequisites: The following are the prerequisites for the operation of the Matyas-Meyer-Oseas hash function:
  - A block-cipher encryption function E shall have been chosen, with a key size that is equal to the block size. The Matyas-Meyer-Oseas hash function has a message digest size that is equal to the block size of the established encryption function. It operates on bit strings of size less than 2^n, where n is the block size, in octets, of the established block-cipher.
  - A fixed representation of integers as binary strings or octet strings shall have been chosen.
- Input: The input to the Matyas-Meyer-Oseas hash function is as follows:
  - A bit string M of size l bits, where $0 \leq l < 2^n$.
- Actions: The hash value shall be derived as follows:
  - Pad the message M according to the following method:
  - Right-concatenate to the message M the binary value consisting of one bit of 1 followed by k bits of 0, where k is the smallest non-negative solution to the equation

    $l + 1 + k \equiv 7n \pmod{8n}$.
  - Form the padded message M by right-concatenating to the resulting string the n-bit string that is equal to the binary representation of the integer l.
  - Parse the padded message M as $M_1 \,||\, M_2 \,||\, \ldots \,||\, M_t$ where each message block M_i is an n-octet string.
  - The output $Hash_t$ is defined by

    $Hash_0 = 0^{8n}$; $Hash_j = E(Hash_{j-1}, M_j) \oplus M_j$ for $j=1,\ldots,t$.

Here, E(K, x) is the ciphertext that results from encryption of the plaintext x, using the established block-cipher encryption function E with key K; the string 0^{8n} is the n-octet all-zero bit string.

- Output: The bit string $Hash_t$ as the hash value.

The cryptographic hash function operates on bit strength of size less than 2^n bits, where n is the block size (or key size) of the established block cipher, in octets. For example, the Matyas-Meyer-Oseas hash function with AES-128 operates on bit strings of size less than 2^{16} bits. It is assumed that all hash function calls are on bit strings of size less than 2^n bits. Any scheme attempting to call the hash function on a bit string exceeding 2^n bits shall output invalid and stop.

H.10 Elliptic curve cryptography manual certificate scheme

H.10.1 Overview

A manual certificate scheme based on elliptic curve cryptography (ECC) that is used in this standard is described.

The manual certificate scheme is used by three entities: a certificate authority CA, a certificate requester U, and a certificate processor V, where U wishes to obtain a manual certificate from CA in order to convey U's associated public key to V.

The manual certificate scheme is described in terms of a certificate generation transformation and a certificate processing transformation. CA, U, and V use these schemes when they wish to communicate.

Prior to use of the scheme, U, V, and CA agree on the parameters with which the scheme shall be used. In particular, this includes U and V obtaining an authentic copy of CA's unique identifier.

CA executes the manual certificate generation transformation to compute an elliptic curve public-key pair for U and a manual certificate MC for this public key provided by CA. V executes the manual certificate processing transformation, to obtain U's purported static public key from U's purported manual certificate MC presented to V.

The manual certificate generation transformation yields a public-key pair and a certificate for this public key. This public-key pair shall be communicated to the purported holder in a secure and authentic way. The mechanism by which this public-key pair is communicated is outside the scope of this standard.

The manual certificate processing transformation yields a static public key (and associated keying information) purportedly bound to the claimed holder; evidence that this public key is genuinely bound to this entity can, however, not be corroborated via processing of the manual certificate. Thus, with manual certificates, the binding of an entity and its public or private key cannot be verified, although one may obtain evidence that some entity that claims to be bound to the public key has indeed access to the corresponding private key, during cryptographic usage of the public key (e.g., via execution of an authenticated key agreement scheme or a signing transformation involving this public-key pair).

The manual certificate generation transformation is specified in H.10.2 and the manual certificate processing transformation is specified in H.10.3.

The prerequisites for the use of the scheme are:

- An infrastructure shall have been established for the operation of the scheme, including a certificate format, certificate processing rules, and unique identifiers. For an example of such an infrastructure, see IETF RFC 3280.
- Each entity has an authentic copy of the system's elliptic curve domain parameters D=(p,a,b,G,n,h) or D=(m,f(x),a,b,G,n,h). These parameters shall have been generated using the parameter generation primitives in SEC 1:2009, 3.1.1.1 or 3.1.2.1. Furthermore, the parameters shall have been validated using the parameter validation primitives in SEC1:2009, 3.1.1.2 or 3.1.2.2.

- Each entity shall be bound to a unique identifier (e.g., distinguished names). All identifiers shall be bit strings of the same size, entityIdSize. Entity U's identifier will be denoted by the bit string U. Entity V's identifier will be denoted by the bit string V. Entity CA's identifier will be denoted by the bit string CA.
- A cryptographic hash function Hash shall have been chosen for use with the ECQV implicit certificate generation scheme. Let hashlen denote the size in bits of the output value of this hash function.
- Each entity shall have decided how to represent elliptic curve points as octet strings (i.e., compressed form, uncompressed form, or hybrid form).
- A fixed representation of octets as binary strings shall have been chosen (e.g., most-significant-bit-first order or least-significant-bit-first order).

H.10.2 Elliptic curve cryptography manual certificate generation transformation

A CA shall execute the following transformation to provide a manual certificate for the user, U. The CA shall obtain an authentic copy of U's identifier.

- Inputs: This routine does not take any inputs.
- Ingredients: The certificate generation transformation employs the key pair generation primitive in SEC 1:2009, 3.2.1, and the manual certificate generation primitive of the established infrastructure.
- Actions: The CA shall proceed as follows:
  - The key pair generation primitive specified in SEC 1:2009, 3.2.1, shall be used to generate an ephemeral key pair (w_U, W_U) for the parameters D.
  - The elliptic curve point W_U shall be converted to the octet string WE_U as specified in SEC 1:2009, 2.3.3.
  - The octet string I_U, which is the to-be-conveyed-manual-certificate data. I_U shall be constructed to contain identification information according to the procedures of the established infrastructure and may also contain other information, such as the intended use of the public key, the serial number of the manual certificate, and the validity period of the manual certificate. The exact form of I_U depends on the manual certificate format specified during the setup procedure.
  - The octet string MC_U, which is U's manual certificate, shall be constructed according to the procedures of the established infrastructure. MC_U shall contain the octet strings I_U and WEU encoded in a reversible manner. The exact form of MC_U depends on the manual certificate format specified during the setup procedure.
- Output: MC_U, which shall serve as U's manual certificate provided by CA.

H.10.3 Elliptic curve cryptography manual certificate processing transformation

V shall execute the following transformation to obtain U's purported static public key from U's purported manual certificate provided by CA. V shall obtain an authentic copy of U's and CA's identifier.

- Input: U's purported manual certificate MC_U provided by CA.
- Ingredients: The manual certificate processing transformation employs the public key validation primitive in SEC 1:2009, 3.2.2, and the manual certificate validation primitive of the established infrastructure.
- Actions: V proceeds as follows:
  - Verify the content of MCU according to the established infrastructure. This includes verifying the contents of the certificate, such as the subject's name and the validity period. If the subject's name is not U, output invalid and stop.
  - Derive I_U from MCU, according to the manual certificate format specified during the setup procedure.

  - Derive CA's identifier from I_U, according to the certificate format specified during the setup procedure. If CA's identifier is unknown to V, output invalid and stop.
  - Derive WE_U from MCU, according to the manual certificate format specified during the setup procedure.
  - Convert the octet string WE_U to the elliptic curve point W_U as specified in SEC 1:2009, 2.3.4
  - Verify that W_U is a valid key for the parameters D as specified in SEC 1:2009, 3.2.2. If the validation primitive rejects the key, output invalid and stop.

- Output: If any of the above verifications has failed, then output invalid and stop; otherwise, output valid and accept W_U as U's purported static public key. (V may accept W_U as U's genuine static public key provided U evidences knowledge to V of the corresponding private key w_U and provided V accepts U to be the only party that may have access to this private key.)

Annex I
(informative)

Definition templates

I.1 Object type template

It is recommended that standard objects and their associated standard object identifiers be identified in a table for quick reference, as shown in Table I.1. This indicates the information needed to define standard object types defined by this standard.

Table I.1 – Table of standard object types

Defining organization:			
Standard object type name **(not expected to be transmitted, size not specified – check DD limits)**	**Standard object type identifier (non-negative)**	**Standard object identifier (non-negative), if applicable** **Used for mandatory objects with exactly one instance per device**	**Object description** **(not expected to be transmitted, size not specified – check DD limits)**
…	…	…	…

Elements of the table include:

- Standard object type name defines the name of the object.
- Standard object type identifier is the standard non-negative numeric identifier of the object type; uniquely identifies this object type.
- Standard object identifier, for standard object types that are required by a device and that may only be instantiated once, represents the standard non-negative numeric identifier for the object instance. This identifier is common to all devices. If 7 bits do not suffice, the high-order bit of the first octet shall be set, and another octet shall be available to extend the value of the identifier.
- Object description is a description of the purpose and intent of this object.

I.2 Standard object attributes template

The template shown in Table I.2 indicates the information needed to define the attributes of a standard object.

Table I.2 – Template for standard object attributes

Standard object type name:				
Standard object type identifier:				
Defining organization:				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of behavior of attribute**
ObjectIdentifier	Key identifier	Unique identifier for the object	Type: Unsigned16.	N/A
			Classification: Static	
			Valid range: 1..32 767	
…	…	…	Type: …	…
			Classification: …	
			Accessibility: …	
			Default value: …	
			Valid range: …	
Reserved for future use	—	—	—	—

Elements of the table include:

- Standard object type name defines the name of the object type.
- Standard object type identifier is the standard numeric identifier of the object type that uniquely identifies this object type. The value of this identifier fits into at most two octets.
- Defining organization is the organization defining this object (e.g., base standard, standard defined extension to the base standard object, industry specific profile (and which industry), special interest group (and which interest group)), or a device vendor.
- Attribute name defines the name of the attribute.
- Attribute ID is the standard numeric identifier of the attribute. All attributes of an object are uniquely identified. If 7 bits do not suffice, the high-order bit of the first octet shall be set, and another octet shall be available to extend the value of the identifier.
- Description is the description of the attribute.
- Type is the data type of the attribute. If the data may vary in size (such as for a variable size OctetString or a variable size VisibleString), then the maximum number of octets of data is indicated.
- Classification is the data classification (constant, static, static-volatile, dynamic, non-cacheable) of the attribute.
- Accessibility is how the attribute may be accessed remotely (e.g., read only, or read/write)
- Initial default value specifies the initial default value.
- Valid value set specifies the valid set of values for this attribute.

I.3 Standard object methods

The template shown in Table I.3 indicates the information needed to describe the methods of a standard object.

Table I.3 – Template for standard object methods

<table>
<tr><td colspan="5">Standard object type name:</td></tr>
<tr><td colspan="5">Standard object type identifier:</td></tr>
<tr><td colspan="5">Defining organization:</td></tr>
<tr><td>Method name</td><td>Method ID</td><td colspan="3">Method description</td></tr>
<tr><td rowspan="6"><name of method></td><td colspan="4">Input arguments</td></tr>
<tr><td>Argument number</td><td>Argument name</td><td>Argument type (data type and size)</td><td>Argument description</td></tr>
<tr><td>…</td><td>…</td><td>…</td><td>…</td></tr>
<tr><td colspan="4">Output arguments</td></tr>
<tr><td>Argument number</td><td>Argument name</td><td>Argument type (data type and size)</td><td>Argument description</td></tr>
<tr><td>…</td><td>…</td><td>…</td><td>…</td></tr>
</table>

Elements of the table include:

- Standard object type name defines the name of the object.
- Standard object type identifier is the standard numeric identifier of the object type that uniquely identifies this object type. The value of this identifier fits into at most two octets.
- Defining organization is the organization defining this object (e.g., base standard, standard defined extension to the base standard object, industry specific profile (and which industry), special interest group (and which interest group)).
- Method name is the name of the method.
- Method ID is the numeric identification of the method. All methods of an object will have unique method identifiers. If 7 bits do not suffice, the high-order bit of the first octet shall be set, and another octet shall be available to extend the value of the identifier.
- Method description is the description of the method.
- List of input parameters and their data types is a list of input parameters, their type and size (if not explicitly discernable from the type), and a description of use (how they are used when sent on a method invoke). These should be listed in order of transmission.

NOTE 1 For simplicity, all parameters are specified. If there are situations where parameters vary, separate methods are appropriate to accommodate each class of variance.

- List of output parameters and their data types is a list of output parameters, their type and size (if not explicitly discernable from the type), and a description of use (how they are used when sent on a method invoke). These should be listed in order of transmission.

NOTE 2 See NOTE 1.

- Description of behavior describes the behavior of the object when this method is invoked.

I.4 Standard object alert reporting template

The template shown in Table I.4 indicates the information needed to describe the alert reporting behavior of a standard object.

Table I.4 – Template for standard object alert reporting

Standard object type name(s):					
Standard object type identifier:					
Defining organization:					
Description of the alert:					
Alert class (Enumerated: alarm or event)	**Alert category (Enumerated: device diagnostic, comm. diagnostic, security, or process)**	**Alert type (Enumerated: based on alert category)**	**Alert priority**	**Value data type**	**Description of value included with alert**
<name of alert>	…	…	…	Type: …	…
				Default value: …	…
				Valid range: …	…

Elements of the table include:

- Standard object type name defines the name of the object.
- Standard object type identifier is the standard numeric identifier of the object type that uniquely identifies this(these) object type(s) that may report this alert. The value of this identifier fits into at most two octets.
- Defining organization is the organization defining this object (e.g., base standard, industry specific profile (and which industry), special interest group (and which interest group)).
- Description of the alert describes the semantic meaning of the alert.
- Alert class indicates if this is an event (stateless) or alarm (state-oriented) type of alert.
- Alert category indicates if this is a device related (e.g., a device specific diagnostic), communication related, security related, or process related alert. Only one category applies. Selection of the best fit for an alert may need to be discussed in order to be best established.
- Alert type is dependent on the alert category. See the alert reporting model in 12.8 for further details.
- Alert priority is the priority of the alert.
- Value and size are the size and value included in the alert report.
- Description of value included in alert report is the description of the value, if a value is included in the alert report (e.g., for a process alarm that is a high alarm, this may be the process variable (PV)).
- Accessibility defines if the attribute is readable, writeable, or both.
- Initial default value indicates the initial default value of the attribute.
- Description of value set describes the set of values that may be taken on by this attribute.
- Description of behavior describes the behavior of this attribute (e.g., when a particular value is written, or error conditions). Restrictions on use (e.g., operators should not write to this attribute) may be noted here.

I.5 Data structure definition

The template for describing data structures that are used to define special data types is given in Table I.5.

Table I.5 – Template for data structures

Standard data type name:		
Defining organization:		
Element name	**Element identifier**	**Element scalar type**
...	...	Type: ...
		Size: ...
		Classification: ...
		Accessibility: ...
		Default value: ...
		Valid range: ...

Annex J
(informative)

Operations on attributes

J.1 Operations on attributes

J.1.1 General

Attribute classification and accessibility dictate the operations permitted on a given attribute. Attribute classification and accessibility are described in 12.6.

J.1.2 Attribute classification

For a discussion of attribute classification, see 12.6.3.

J.1.3 Retrieving, setting, and resetting attributes

J.1.3.1 General

Attributes defined in the management objects can be accessed using the standard ASL-provided read or write services. Such operations enable configuration of the layers, as well as monitoring of their status. They can be used to retrieve, set / modify, and reset the values of attributes. The service primitives for these services, as well as the enumerated service feedback codes, are given in Clause 12.

Attributes can be reset using the write service by writing the default value to the relevant attribute. If a reset attribute is defined for a management object, it can be used to reset all the attributes in that management object that belong to certain classes of attributes.

More complex methods may be defined if necessary, but only if the equivalent results cannot be achieved using the more direct read / write services. A complex method may be warranted, for example, to replace a sequence of communication transactions in order to save energy. A complex method may also be warranted when synchronization issues may result if individual actions are used, rather than an atomic transaction set.

J.1.3.2 Scheduled operations to enable synchronized cutover

The generic method template Scheduled_Write provided in Table J.1 can be used to define a method for writing a value to an attribute at a scheduled TAI time. It can also be used to reset an attribute to its default value at a scheduled TAI time.

Table J.1 – Scheduled_Write method template

<table>
<tr><th>Method name</th><th>Method ID</th><th colspan="3">Method description</th></tr>
<tr><td rowspan="10">Scheduled_Write</td><td><given in management object definition></td><td colspan="3">Method to write a value to an indicated attribute at an indicated TAI time</td></tr>
<tr><td colspan="4">Input arguments</td></tr>
<tr><td>Argument number</td><td>Argument name</td><td>Argument type (data type and size)</td><td>Argument description</td></tr>
<tr><td>1</td><td>Attribute_ID</td><td>Data type: Unsigned16 <given in management object definition></td><td>The attribute ID in the management object to which this method is being applied</td></tr>
<tr><td>2</td><td>Scheduled_TAI_Time</td><td>Data type: TAITimeRounded</td><td>TAI time at which the value should be written to the attribute</td></tr>
<tr><td>3</td><td>Value</td><td>Data type: <given in management object definition></td><td>The value that needs to be written to the attribute</td></tr>
<tr><td colspan="4">Output arguments</td></tr>
<tr><td>Argument number</td><td>Argument name</td><td>Argument type (data type and size)</td><td>Argument description</td></tr>
<tr><td colspan="4">None</td></tr>
</table>

The service feedback codes given in 12.17.4.2.2 are expected to be used to indicate if the method execution was successful or not. If not successful, this code provides information indicating why it was not successful.

J.1.4 Retrieving and setting structured attributes

The generic method templates Read_Row and Write_Row given in Table J.2 and Table J.3 can be used for defining methods that retrieve and set/modify the values of structured attributes. When the structured attribute is visualized as an information table, these methods allow access to a particular row based on one or more unique index field values. It is assumed that each table has at least one unique index field. The index field may either be a single element or the concatenation of a few elements in the row.

The input argument Scheduled_TAI_Time in the Write_Row method template allows scheduled operation for a particular row of the structured attribute. A value of 0 for this argument indicates an immediate write operation.

Table J.2 – Read_Row method template

Method name	Method ID	Method description		
Read_Row	<given in management object definition>	Method to read the value of a single row of a structured attribute whose data is visualized as an information table		
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Attribute_ID	Data type: Unsigned16 <given in management object definition>	The attribute ID in the management object to which this method is being applied
	2	Index_1	Data type of first index field of the structured attribute <given in management object definition>	The first index field in the structured attribute to access a particular row
	n+1	Index_n	Data type of n^{th} index field of the structured attribute <given in management object definition>	The n^{th} index field in the structured attribute to access a particular row
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Data_Value	Data type: <given in management object definition>	An octet string that contains the data value

Table J.3 – Write_Row method template

Method name	Method ID	Method description		
Write_Row	<given in management object definition>	Method to set/modify the value of a single row of a structured attribute whose data is visualized as an information table		
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Attribute_ID	Data type: Unsigned16 <given in management object definition>	The attribute ID in the management object to which this method is being applied
	2	Scheduled_TAI_Time	Data type: TAITimeRounded	TAI time at which the value should be written to the row of the structured attribute
	3	Index_1	Data type of first index field of the structured attribute <given in management object definition>	The first index field in the structured attribute to access a particular row
	N+2	Index_n	Data type of n^{th} index field of the structured attribute <given in management object definition>	The n^{th} index field in the structured attribute to access a particular row
	N+3	Data_Value	Data type: <given in management object definition>	An octet string that contains the data value
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	None			

The service feedback codes given in 12.17.4.2.2 are expected to be used to indicate if the method execution was successful or not. If not successful, this code provides information indicating why it was not successful.

A method based on the Write_Row template can also be used to create a new row in the structured attribute if the index field(s) provided in the input argument(s) does(do) not exist.

J.1.5 Resetting structured attribute values

For a structured attribute, the generic method template Reset_Row given in Table J.4 can be used for defining methods that reset/clear certain values in the structured attribute. The input argument Scheduled_TAI_Time in this method allows a scheduled reset operation. A value of 0 for this argument indicates an immediate reset operation.

Table J.4 – Reset_Row method template

Method name	Method ID	Method description		
Reset_Row	<given in management object definition>	Method to reset a single row of a structured attribute whose data is visualized as an information table		
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Attribute_ID	Data type: Unsigned16 <given in management object definition>	The attribute ID in the management object to which this method is being applied
	2	Scheduled_TAI_Time	Data type: TAITimeRounded	TAI time at which the row of the structured attribute should be reset
	3	Index_1	Data type of first index field of the structured attribute <given in management object definition>	The first index field in the structured attribute to access a particular row
	n+2	Index_n	Data type of n^{th} index field of the structured attribute <given in management object definition>	The n^{th} index field in the structured attribute to access a particular row
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	None			

The service feedback codes given in 12.17.4.2.2 are expected to be used to indicate if the method execution was successful or not. If not successful, this code provides information indicating why it was not successful.

J.1.6 Deleting structured attribute values

The generic method template Delete_Row described in Table J.5 can be used for defining methods that delete the values of structured attributes. The input argument Scheduled_TAI_Time in this method allows a scheduled delete operation. A value of 0 for this argument indicates an immediate delete operation.

Table J.5 – Delete_Row method template

Method name	Method ID	Method description		
Delete_Row	<given in management object definition>	Method to delete a single row of a structured attribute whose data is visualized as an information table		
	Input arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	1	Attribute_ID	Data type: Unsigned16 given in management object definition>	The attribute ID in the management object to which this method is being applied
	2	Scheduled_TAI_Time	Data type: TAITimeRounded	TAI time at which the row of the structured attribute should be deleted
	3	Index_1	Data type of first index field of the structured attribute <given in management object definition>	The first index field in the structured attribute to access a particular row
	n+2	Index_n	Data type of n^{th} index field of the structured attribute <given in management object definition>	The n^{th} index field in the structured attribute to access a particular row
	Output arguments			
	Argument number	**Argument name**	**Argument type (data type and size)**	**Argument description**
	None			

The service feedback codes given in 12.17.4.2.2 are expected to be used to indicate if the method execution was successful or not. If not successful, this code provides information indicating why it was not successful.

J.2 Synchronized cutover

A synchronized cutover capability is needed for some attributes and structured attributes that represent management information. For such an attribute, updates for the attribute value may be scheduled by indicating the TAI cutover time information; this operation may be accomplished by using one of the methods defined above. Such updates are sent to the management object for which this attribute is defined. The management object immediately validates whether the cutover is feasible, and, if feasible, arranges for the cutover to occur on schedule.

Annex K
(normative)

Standard object types

Annex K specifies the standard object types defined by this standard. Each object type has three pieces of information to identify it:

- a corresponding standard object type identifier that identifies the standard defined base object type (example: analog input);
- a corresponding object standard subtype identifier that identifies the standard subtype of a standard base type (example: analog input specialized for temperature); and
- a corresponding vendor subtype identifier that identifies a vendor specific subtype of either a standard base object or standard subtype.

Standard base objects shall have their object subtype identifier value equal to zero (0) and their vendor subtype identifier equal to zero (0).

A newer version of this standard that finds it necessary to extend the base object type definition of this standard may maintain the standard object identifier value and the subtype value of zero (0). This is permitted since the DMO contains an attribute to represent the version of the standard in use by the device, which can thus be used to establish the base object type structure in use.

IEC62734 industry profiles may define a standard object subtype as a standard object. Doing so creates a standard profile subtype. This standard provides a range of 1..255 to represent all such standard object subtype across all profiles.

Vendors may also subtype either a standard base object or a standard subtype object. This standard provides a range of 1..255 for vendor specific subtyping.

Object subtyping occurs when:

- one or more attributes is/are added to the base type;
- one or more methods is/are added to the base type;
- one or more alerts is/are added to the base type; or
- any combination of the above.

Examples of object identification with subtyping follow:

- Analog input standard base object:
  - object type identifier = 99,
  - object standard subtype identifier = 0,
  - vendor subtype identifier = 0.
- Analog input temperature subtype object:
  - object type identifier = 99,
  - object standard subtype identifier = (this standard defines (this standard's profile team), range 1..255),
  - vendor subtype identifier = 0.
- Vendor-specific analog input object:
  - object standard type identifier = 99,
  - object standard subtype identifier = 0,

  - vendor-specific subtype identifier = (vendor defines, range 1..255).
- Vendor-specific analog input temperature object:
  - object standard type identifier = 99,
  - object standard subtype identifier = n,
  - vendor specific subtype identifier = (vendor defines, range 1..255).

Table K.1 specifies standard object types.

Table K.1 – Standard object types

Object type	Standard object type identifier (1 octet)	Standard object industry subtype identifier (1 octet)	Object identifier restrictions
Object types available to all UAPs			
Null object	0	0	Reserved
UAP management object	1	0	This object is required for all UAPs, but is not required for the DMAP
AlertReceiving object	2	0	
UploadDownload object	3	0	
Concentrator object	4	0	
Dispersion object	5	0	
Tunnel object	6	0	
Interface object	7	0	
Reserved for use by this standard for standard UAP objects	8..50	0	Reserved for future standard object definitions for profile independent objects
Reserved for use by this standard	51..95	0	Industry-specific types
Process control industry object types			
Analog input	99	0	Analog input
Analog output	98	0	Analog output
Binary input	97	0	Binary input
Binary output	96	0	Binary output
DMAP object types			
DMAP: Device management object (DMO)	127	0	This object facilitates the management of the device's general device-wide functions
DMAP: Alert reporting management object (ARMO)	126	0	This object facilitates the management of the device's alert reporting functions
DMAP: Device security management object (DSMO)	125	0	This object facilitates the management of the device's security functions
DMAP: DL management object (DLMO)	124	0	This object facilitates the management of the device's DL
DMAP: NL management object (NLMO)	123	0	This object facilitates the management of the device's NL

Object type	Standard object type identifier (1 octet)	Standard object industry subtype identifier (1 octet)	Object identifier restrictions
DMAP: TL management object (TLMO)	122	0	This object facilitates the management of the device's TL
DMAP: Application sublayer management object (ASLMO)	121	0	This object facilitates the management of the device's application sublayer
DMAP: Device provisioning object (DPO)	120	0	This object facilitates the provisioning of the device before it joins the network
DMAP: Health reports concentrator object (HRCO)	128	0	This object facilitates the periodic publication of device health reports to the system manager
Standard management objects	119..114	0	
System time service object (STSO)	100	0	This object facilitates the management of system-wide time information
Directory service object (DSO)	101	0	This object facilitates the management of addresses for all existing devices in the network
System configuration object (SCO)	102	0	This object facilitates the configuration of the system including contract establishment, modification and termination
Device management service object (DMSO)	103	0	This object facilitates device joining, device leaving, and device configuration
System monitoring object (SMO)	104	0	This object facilitates the monitoring of system performance
Proxy security management object (PSMO)	105	0	This object acts as a proxy for the security manager
Device provisioning service object (DPSO)	106	0	This object facilitates device provisioning
Standard system management objects	107..113	0	Reserved for standard management object type definitions. See Clause 6 for details
Vendor-defined types			
Vendor-defined objects	129..255	0	Reserved for use by implementers

Table K.2 specifies standard object instances.

Table K.2 – Standard object instances

Object type	Standard object type identifier (1 octet)	Standard object industry subtype identifier (1 octet)	Standard object identifier (1 octet)	Object identifier restrictions
Object types available to all UAPs				
Null object	0	0	0	Reserved
UAP management object	1	0	1	This object is required for all UAPs, but is not required for the DMAP
UploadDownload object	3	0	2	For UAP upgrade use only
Process control industry object types				
N/A				
DMAP object types				
DMAP: Device management object (DMO)	127	0	1	This object facilitates the management of the device's general device-wide functions
DMAP: Alert reporting management object (ARMO)	126	0	2	This object facilitates the management of the device's alert reporting functions
DMAP: Device security management object (DSMO)	125	0	3	This object facilitates the management of the device's security functions
DMAP: DL management object (DLMO)	124	0	4	This object facilitates the management of the device's DL
DMAP: NL management object (NLMO)	123	0	5	This object facilitates the management of the device's NL
DMAP: TL management object (TLMO)	122	0	6	This object facilitates the management of the device's TL
DMAP: Application sublayer management object (ASLMO)	121	0	7	This object facilitates the management of the device's application sublayer
DMAP: Upload/download object (UDO)	3	0	8	This object facilitates the management of the device's upload/download functions
DMAP: Device provisioning object (DPO)	120	0	9	This object facilitates the provisioning of the device before it joins the network

Object type	Standard object type identifier (1 octet)	Standard object industry subtype identifier (1 octet)	Standard object identifier (1 octet)	Object identifier restrictions
DMAP: Health reports concentrator object (HRCO)	128	0	10	This object facilitates the periodic publication of device health reports to the system manager
System management AP standard types				
System time service object (STSO)	100	0	1	This object facilitates the management of system-wide time information
Directory service object (DSO)	101	0	2	This object facilitates the management of addresses for all existing devices in the network
System configuration object (SCO)	102	0	3	This object facilitates the configuration of the system including contract establishment, modification and termination
Device management service object (DMSO)	103	0	4	This object facilitates device joining, device leaving, and device configuration
System monitoring object (SMO)	104	0	5	This object facilitates the monitoring of system performance
Proxy security management object (PSMO)	105	0	6	This object acts as a proxy for the security manager
Upload/download object (UDO)	3	0	7	This object facilitates downloading firmware/data to devices and uploading data from devices
Alert-receiving object (ARO)	2	0	8	This object receives all the alerts destined for the system manager
Device provisioning service object (DPSO)	106	0	9	This object facilitates device provisioning
Health reports concentrator object (HRCO)	4	0	10	This object facilitates the periodic publication of device health reports to the system manager
Vendor-defined types				
…	…	…	…	…

Annex L
(informative)

Standard data types

Table L.1 specifies the standard data type identifiers for the standard data types. Standard data types are defined for constructs that are accessible using ASL services, such as read and write.

NOTE 1 It is possible for data structures to not be directly accessible using ASL services, e.g., a data structure that is used as a parameter of a method, but which is not exposed as an ASL-accessible object attribute.

NOTE 2 Many of the type identifiers in this table are based on type identifiers used in an existing IEC standard.

Table L.1 – Standard data types

Data type	Type identifier (Unsigned16)	Size (octets)
Reserved types		
Invalid (type not specified)	0	0
AP standard data structure types		
Communication association endpoint	468	See Table 265
ObjectAttributeIndexAndSize	469	See Table 264
Communication contract data	470	See Table 266
Alert communication endpoint	471	See Table 267
ObjectIDandType	472	See Table 271
Unscheduled correspondent	473	See Table 272
Process control types		
Process control value and status for analog value	65	See Table 300
Process control value and status for binary value	66	See Table 301
Process control scaling	68	See Table 304
Process control mode	69	See Table 302
Alert descriptor types		
Process control alarm report descriptor for analog with single reference condition	498	See Table 270
Alert report descriptor (also used for process control binary alarms)	499	See Table 269

Data type	Type identifier (Unsigned16)	Size (octets)
General communication / management types		
Contract_Data	401	See Table 30
Address_Translation_Row	402	See Table 14
New_Device_Contract_Response	405	See Table 31
Metadata_attribute	406	See Table 2
Security_Sym_Join_Request	410	See Table 62
Security_Sym_Join_Response	411	See Table 63
Security_Sym_Confirm	412	See Table 66
Security_Pub_Join_Request	415	See Table 70
Security_Pub_Join_Response	416	See Table 70
Security_Pub_Confirm_Request	417	See Table 72
Security_Pub_Confirm_Response	418	See Table 72
Security_New_Session_Request	420	See Table 81
Security_New_Session_Response	421	See Table 82
Security_Key_and_Policies	422	See Table 84
Security_Key_Update_Status	423	See Table 85
DPSOWhiteListTbl	440	See Table 372
NLContractTbl	441	See Table 207
NLRouteTbl	442	See Table 208
NLATTbl	443	See Table 209

Annex M
(normative)

Identification of tunneled legacy fieldbus protocols

Table M.1 lists the Unsigned8 protocol identification values currently defined to tunnel legacy wired fieldbus protocols via the tunnel object.

Table M.1 – Identification of tunneled legacy fieldbus protocols

Value	Protocol
0	None (required)
1	HART (see IEC 61158)
2	FF-H1 (see IEC 61158)
3	Modbus/RTU (see IEC 61158)
4	PROFIBUS PA (see IEC 61158)
5	CIP (see IEC 61158)
6..255	<reserved>

NOTE These protocol identification values have been isolated into Annex M in order to facilitate ease of maintenance.

Value 0 for None should be preserved or tunnel functionality will be impaired.

Annex N
(informative)

Tunneling and native object mapping

N.1 Overview

Tunneling involves the exchange of PDUs of one protocol by using a second protocol. Most often these PDUs are application PDUs, but lower layer PDUs may also be exchanged. The PDU is encapsulated in the second protocol at an origination node and sent through the network to a termination node. With tunneling, what goes in one end comes out the other end, no more, no less.

Foreign protocol application communication (FPAC) is a more sophisticated PDU exchange mechanism. It involves the usage of additional mechanisms, including caching, compression, address translation, and proxy. As far as the application is concerned, the same PDUs are still exchanged between the origination node and the termination node as with tunneling. The difference is that the additional mechanisms act to improve energy efficiency and host system responsiveness.

N.2 Tunneling

Tunneling carries messages verbatim between endpoints of a tunnel. This standard provides tunneling that uses un-buffered client/server exchange of foreign PDUs between exactly two pre-configured tunnel endpoints. No interpretation of the PDU content is required. For most legacy protocols, this method will not be energy efficient, and some protocols may not operate properly due to variable or lengthy response times associated with sleeping devices. Regardless of the shortcomings, in many cases this will be the most expedient method for adapting existing devices and systems to this standard.

An extension of tunneling interprets the addressing within foreign PDUs to allow dynamic foreign PDU exchange with multiple endpoints.

N.3 Foreign protocol application communication

Tunneling is not an appropriate mechanism for most low-power wireless link applications. It is usually necessary to minimize PhPDU overhead and the number of transactions in order to conserve energy stored in batteries or to operate within the power budget of scavenging and harvesting techniques. In addition, foreign protocols often have a need for fast response in order to avoid built-in timeouts. Devices in low-power wireless operation are most often in a sleep mode and thus cannot respond immediately.

FPAC increases energy efficiency and addresses potential timing issues by using change-of-state transfer and caching to eliminate redundant transfer. Improvements in energy efficiency and performance are achieved by caching the information in the gateway, transferring information to the gateway only when it changes, and providing a heartbeat mechanism for integrity. This minimizes transfers initiated by the end devices (i.e., periodic publications), as well as minimizing transfers initiated by the foreign communication link (i.e., multi-master access through the gateway). In addition, this method can address foreign protocol timing requirements. Compared to tunneling, additional effort is necessary to translate the foreign protocol.

This standard provides support for FPAC that minimizes PhPDU overhead using a combination of techniques:

- Encapsulation is limited to a single encapsulation. Protocol translators provide additional encapsulation across foreign links as necessary.
- Encapsulation is achieved through configuration agreement by carrying the foreign protocol within the protocol defined by this standard, rather than by carrying additional protocol headers. Mapping occurs as follows:
  - Transport supported relationships (publish/subscribe and client/server).
  - Foreign addresses and native addresses.
  - Size fields and integrity fields.
- This standard provides a native application service format for message exchange. Foreign protocols have their own service formats and message exchange protocols. The tunnel object allows the transfer of foreign APDUs with no extraneous overhead imposed by the native application service format.

This standard provides support for FPAC that minimizes transaction overhead using the following techniques:

- Distributed buffer caching mechanisms to minimize redundant transfer of unchanged data between gateways and end devices.
- Periodic, change-of-state (CoSt), and aperiodic transfer mechanisms.
- Watchdog timers to monitor endpoint and communication channel availability and assure data quality.

This standard provides support for FPAC that improves foreign protocol device access timing performance (and minimizes unnecessary transactions) by the provision of buffered device information through a gateway high side interface.

NOTE Change of state (CoSt) is distinct from class of service (CoS) as defined by IEEE 802.1Q.

N.4 Native object mapping

This standard supports a native object format and messaging services. Automation-specific objects can be used to support protocol translation by using these objects to perform a mapping of the foreign protocol into these objects and their messaging. Compared to the tunneling and FPAC methods, additional effort is necessary to translate the foreign protocol.

N.5 Tunneling and native object mapping tradeoffs

Native object mapping has a unique advantage in the ability to build a single standard-compliant end device for use with multiple foreign protocols. This is especially attractive for new devices.

Tunneling and FPAC have an advantage in simplicity for adapting wired automation devices through an adapter. Little, if any, translation may be required on either end.

Using tunneling in conjunction with native object mapping is also useful. This allows common legacy functions to use native object mapping, while rarely used functions can be tunneled. This can lead to less total effort in protocol translation.

Annex O
(informative)

Generic protocol translation

O.1 Overview

This standard does not include protocol translators. It does include features to support the construction of protocol translators (generally located within gateways) for common fieldbus protocols, where such a translator would also be sensitive to the constraints of low-power wireless automation networks. Since specific protocol translators are not defined in this standard, all support for protocol translation is thus generic.

Annex O provides an example of how to use the tunnel object and a conceptual GIAP to support common protocol translation interactions. The tunnel object includes the normative features to support protocol translation.

Specific protocol translators (for specific fieldbuses) could include Annex O, potentially in modified form. They could also use a different approach. Such choices are not specified by this standard.

O.2 Publish

A portion of a generic gateway is depicted in Figure O.1, which relates to the usage of publication. A generic protocol translator interacts with a gateway UAP through the GIAP. The gateway UAP uses the TUN object to interact with remote peers via the lower protocol suite through the ASL SAP.

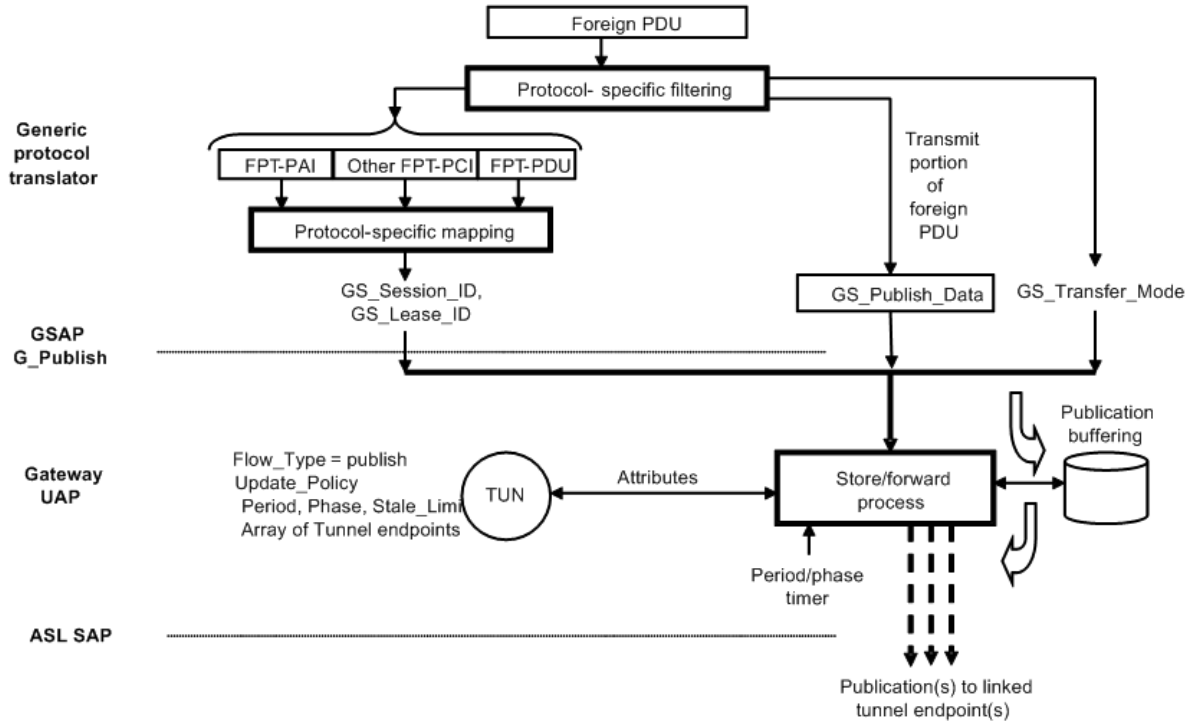


Figure O.1 – Generic protocol translation publish diagram

A foreign PDU is received by the protocol translator, and protocol-specific filtering is applied. Depending on the protocol, a combination of FPT-PAI, other FPT-PCI, and FPT-PDU may be necessary in order to determine the proper GS_Session_ID and GS_Lease_ID for GIAP usage in linking to a subscriber. The protocol-specific filtering determines the portion of the foreign PDU that needs to be transmitted (GS_Publish_Data) and foreign protocol-specific transport parameters such as priority (GS_Transfer_Mode). The parameters are then used to invoke GIAP services.

The GS_Session_ID and GS_Lease_ID are used by the gateway UAP to identify the TUN object and to retrieve the necessary parameters for store and forward processing decisions. GS_Publish_Data is buffered and forwarded at the appropriate time based on the Update_Policy, the period, the phase, the Stale_Limit, and the prior and current data content. Store and forward decisions are also driven by timer events based on the period and the phase. The ASL SAP is used to forward any messages.

A publication may be sent to one or more endpoints depending on the number of elements contained by the array of tunnel endpoints.

O.3 Subscribe

A portion of a generic gateway is depicted in Figure O.2, which relates to the usage of subscription. A generic protocol translator interacts with a gateway UAP through the GIAP. The gateway UAP uses the TUN object to interact with remote peers via the lower protocol suite through the ASL SAP.

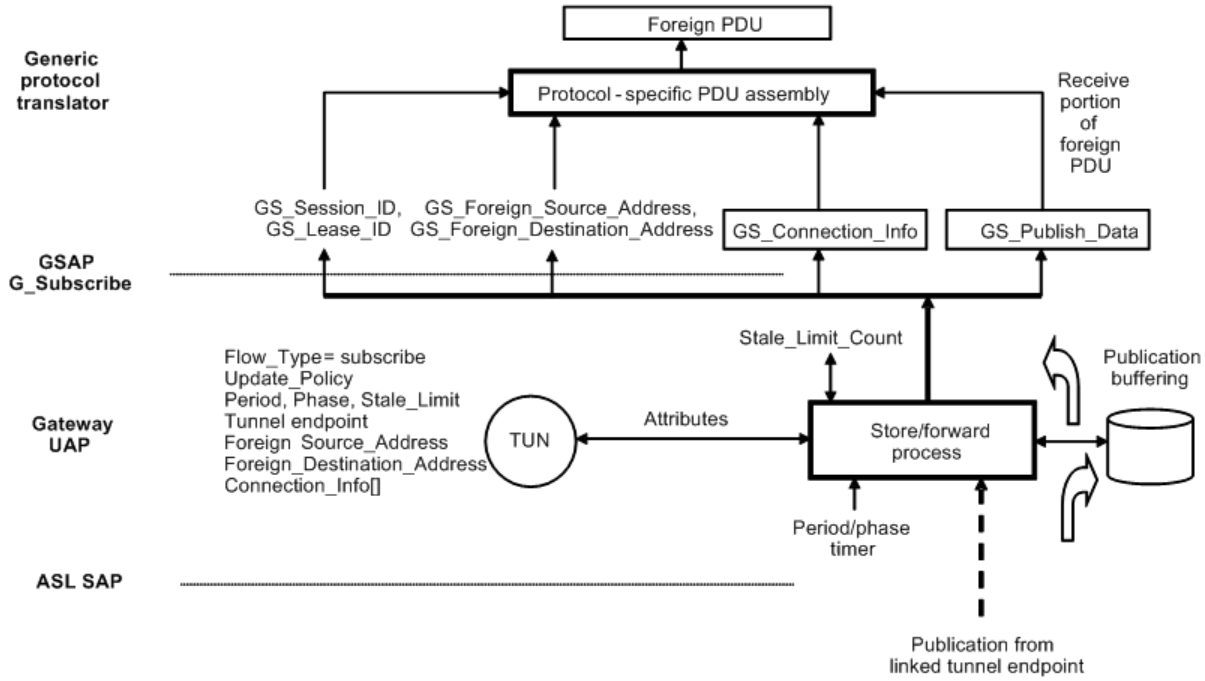


Figure O.2 – Generic protocol translation subscribe diagram

A publication APDU arrives at the gateway UAP through the ASL SAP. The addressing indicates a local TUN object that is linked to the remote publisher TUN object. The necessary attributes are retrieved from the TUN object for store and forward decisions.

Publication data includes the GS_Publish_Data from the publisher. Publication buffering is based on Update_Policy, the Period, the Phase, the Stale_Limit, and Period/Phase based timer events. Forwarding occurs to the protocol translator through the GIAP based on polled and event driven interaction with the protocol translator. The gateway UAP also stores and includes the GS_Session_ID and GS_Lease_ID for the protocol translator to identify the

publication. Publication specific information may be stored locally and used to reduce unnecessary transmission of the information. This information includes addressing information (GS_Foreign_Source_Address and GS_Foreign_Destination_Address) and connection specific information (GS_Connection_Info).

The protocol translator performs a protocol-specific assembly to generate the foreign PDU.

O.4 Client

A portion of a generic gateway is depicted in Figure O.3, which relates to the transmission of client/server tunneled messages. A generic protocol translator interacts with a gateway UAP through the GIAP. The gateway UAP uses the TUN object to interact with remote peers via the lower protocol suite through the ASL SAP.

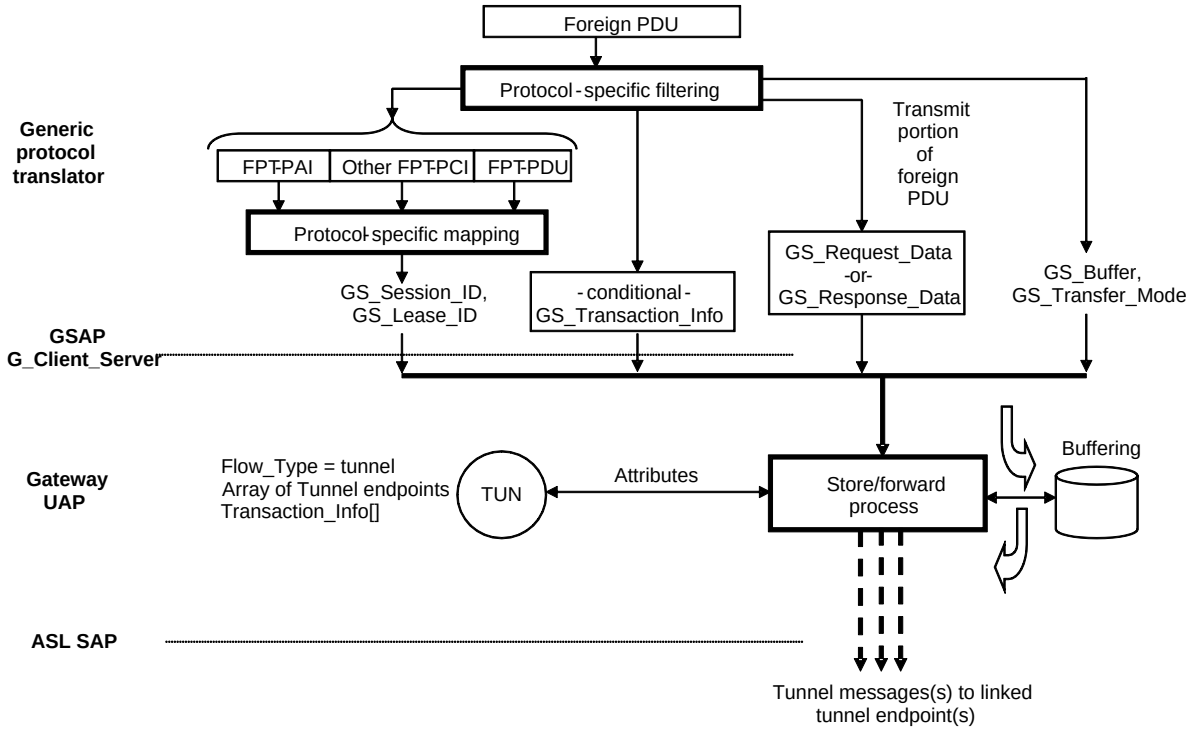


Figure O.3 – Generic protocol translation client/server transmission diagram

A foreign PDU is received by the protocol translator, and protocol-specific filtering is applied. Depending on the protocol, a combination of FPT-PAI, other FPT-PCI, and FPT-PDU may be necessary in order to determine the proper GS_Session_ID and GS_Lease_ID for GIAP usage. Protocol-specific filtering determines the portion of the foreign PDU that needs to be transmitted (GS_Request_Data or GS_Response_Data) and the appropriate transport parameters such as priority (GS_Transfer_Mode). For requests, the SDU may also specify GS_Transaction_Info that is to be returned at the GIAP when a matching response arrives. The parameters are then used to invoke GIAP services.

The GS_Session_ID and GS_Lease_ID are used by the gateway UAP to identify the TUN object and to retrieve the necessary parameters for store and forward processing decisions. The GIAP information (GS_Request_Data or GS_Response_Data) may be buffered before forwarding, depending on whether buffering is requested (GS_Buffer), depending on the prior buffer content, and depending on whether a request or response is specified. The ASL SAP is used to forward any messages.

A tunnel request message may be sent to one or more endpoints depending on the number of elements contained by the array of tunnel endpoints. A tunnel response message can be sent to a single endpoint, but multiple responses can be sent to the same endpoint over time.

O.5 Server

A portion of a generic gateway is depicted in Figure O.4, which relates to the reception of client/server tunneled messages. A generic protocol translator interacts with a gateway UAP through the GIAP. The gateway UAP uses the TUN object to interact with remote peers via the lower protocol suite through the ASL SAP.

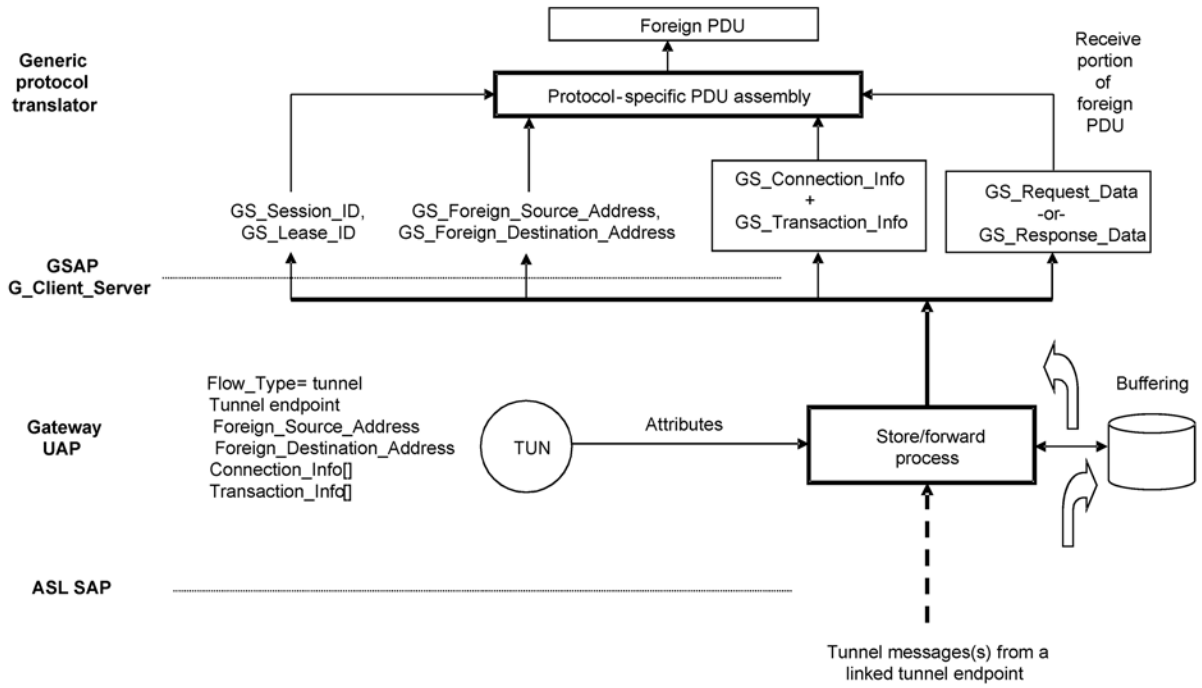


Figure O.4 – Generic protocol translation client/server reception diagram

A tunnel request or response APDU arrives at the gateway UAP through the ASL SAP. The addressing indicates a local TUN object that is linked to a remote TUN object. The necessary attributes are retrieved from the TUN for store and forward decisions.

Tunnel APDU data includes either GS_Request_Data or GS_Response_Data. Depending on the tunnel mode, the response data may be buffered to answer subsequent requests from local buffers. Forwarding occurs to the protocol translator through the GIAP based on polled and event driven interaction with the protocol translator. The gateway UAP also stores and includes the GS_Session_ID and GS_Lease_ID for the protocol translator to identify the tunnel data. Tunnel message specific information may be stored and used to reduce unnecessary duplicated transmission of the information. This includes addressing information (GS_Foreign_Source_Address and GS_Foreign_Destination_Address), connection specific information (GS_Connection_Info) and transaction specific information (GS_Transaction_Info) to be conveyed in responses.

The protocol translator performs a protocol-specific assembly to generate the foreign PDU.

Annex P
(informative)

Exemplary GIAP adaptations for this standard

P.1 General

This standard does not define functionality for a complete gateway. It does include supporting examples that allow gateway construction by the addition of a protocol translator, and a hardware interface and protocol stack for a foreign network. Annex P does not define a protocol translator; that might be a subject for future standardization.

Annex P provides an example of a conceptual interface that would be internal to a gateway – the GIAP, which is intended to be an abstraction of the underlying wireless system. In particular, it is intended to provide an abstraction for the wireless system described in this standard.

Annex P describes one way to implement the informative GIAP by using this standard's normative objects and services. It is not a complete design, but a reference to aid understanding.

Specific gateways (for specific fieldbuses) could include Annex P, thus making it normative. They could also determine a different approach that was compliant.

In this exemplary gateway, the GIAP services are implemented as a specialized UAP that uses native objects as defined in this standard.

P.2 Parameters

GS_Network_Address is the IPv6Address.

P.3 Session

The GIAP session service tracks resources and releases the resource when the session is closed or expires. Resources include communication contracts, bulk transfers in progress, buffered information, publication/subscribe/client/server resources in objects, and alert subscriptions.

P.4 Lease

The GIAP lease service allows allocation of resources and individual release when the lease is closed or expires. Resources include communication contracts and object resources for: bulk transfer, publish/subscribe, client/server, and alerts.

A lease differs from a communication contract in that a lease allocates resources both within a gateway entity and, when needed, the resources corresponding to a related communication contract.

The specification of multiple IPv6Addresses within the GS_Network_Address_List represents a multicast group. Specifying multiple addresses will result in a simulated multicast via multiple unicast operations. Even though this is a single lease, simulated multicast requires the allocation of multiple point-to-point contracts and simultaneous management of this contract set within the gateway.

GS_Resource specifies the bulk transfer item for a lease (Destination_Port and OID). GS_Resource is also used in the linkage of matching sets of TUN objects and matching CON and DIS objects. A matched publisher and subscriber(s) specify related values in lease creation. These values, along with the GS_Network_Address_List, allow the Array of Tunnel endpoint to be filled on linked TUN objects and CON and DIS objects to be allocated and linked.

Subscriber leases specify GS_Update_Policy, GS_Period, GS_Phase, and GS_Stale_Limit.

P.5 Device list report

There is no specific adaptation information for this item.

P.6 Topology report

There is no specific adaptation information for this item.

P.7 Schedule report

There is no specific adaptation information for this item.

P.8 Device health report

There is no specific adaptation information for this item.

P.9 Neighbor health report

GS_Signal_Strength maps to ED and GS_Signal_Quality maps to LQI as defined in 9.1.15.2.

P.10 Network health report

There is no specific adaptation information for this item.

P.11 Time

There is no specific adaptation information for this item.

P.12 Client/server

P.12.1 General

The GIAP client/server service uses the TUN object or the IFO, depending on the lease establishment.

P.12.2 Native access

Where the lease establishment specifies GS_Protocol_Type = 0, the native protocol is configured through an IFO, the GS_Network_Address_List is empty, and the GS_Lease_Parameters specify only GS_Transfer_Mode, which in turn specifies both priority and discard eligibility, as defined in Clause 12 for the read, write and execute services.

The payloads (GS_Request_Data and GS_Response_Data) conform to the native APDU formats and use only the ASL service types: read, write, and execute. The IFO objects in gateways transfer these payloads via the read, write, and execute services. GS_Transfer_Mode is used with each transfer to indicate the quality of service, including priority, associated with the transfer.

GS_Buffer is used to request buffered and unbuffered behavior as appropriate to the ASL service and attribute classifications.

GS_Transaction_Info is empty.

The native client/server service is used to address native objects in the gateway.

P.12.3 Foreign access

Where the lease establishment specifies a GS_Protocol_Type, a foreign protocol is configured through a TUN object. GS_Network_Address_List is supplied to establish the remote TUN endpoints. GS_Resource is used to determine whether 2-part or 4-part tunnel services apply and to match the TUN endpoints within devices. A lone client or server lease establishes a 2-part tunnel. A pair of client and server leases with the same GS_Resource establishes a 4-part tunnel. GS_Lease_Parameters supply GS_Connection_Info on Server services as appropriate for the foreign protocol and GS_Transfer_Mode in order to set default transfer quality of service and priority.

The payloads (GS_Request_Data and GS_Response_Data) conform to the foreign APDU formats, including specification of foreign service types and service-specific fields. The TUN objects in gateways transfer these payloads by using the 2-part and 4-part tunnel services. GS_Cache is used to request buffered and unbuffered behavior as appropriate to the TUN object configuration and the foreign protocol requirements. GS_Transfer_Mode is used with each transfer to indicate the quality of service, including priority, associated with the transfer.

The GS_Transfer_Mode specifies priority and discard eligibility, as defined in Clause 12 for the tunnel service.

GS_Transaction_Info is supplied on client services and returned on server services as appropriate for the foreign protocol.

P.13 Publish/subscribe

P.13.1 General

The GIAP publish/subscribe service uses the TUN object or CON and DIS objects, depending on the lease establishment.

P.13.2 Native access

Where the lease establishment specifies GS_Protocol_Type = 0, the native application protocol will be published through the CON object and subscribed through the DIS object. GS_Network_Address_List is empty. GS_Lease_Parameters contain only GS_Transfer_Mode in order to set default transfer quality of service and priority.

GS_Network_Address_List is used to establish the publish and subscribe endpoints. GS_Network_Address determines the remote device address. GS_Resource is used to determine the DIS object within this device. A local CON object is selected to be linked with the remote DIS object. GS_Lease_Parameters supply GS_Update_Policy, GS_Period, GS_Phase, and GS_Stale_Limit to establish the periodic or changes of state behavior for the CON and DIS objects. GS_Connection_Info is empty.

The publication payload (GS_Publish_Data) is sent and received in NativeIndividualValue or NativeValueList format. The CON and DIS objects in gateways transfer these payloads by using the publish service. GS_Transfer_Mode is provided with each transfer in order to indicate the quality of service, including priority, associated with the transfer.

The GS_Transfer_Mode specifies priority and discard eligibility, as defined in Clause 12 for the publish service.

P.13.3 Foreign access

Where the lease establishment specifies GS_Protocol_Type not equal to 0, GS_Protocol_Type is used to specify the foreign application protocol that will be published through the TUN objects. GS_Network_Address_List is used to establish the remote TUN endpoints. GS_Resource is used to match the TUN endpoints within devices. GS_Lease_Parameters supply GS_Update_Policy, GS_Period, GS_Phase, and GS_Stale_Limit to establish the periodic or changes of state behavior for Publish and Subscribe services. GS_Lease_Parameters supply GS_Connection_Info on Subscribe services as appropriate for the foreign protocol and GS_Transfer_Mode in order to set default transfer quality of service and priority.

The publication payload (GS_Publish_Data) is sent and received in non-native format. The TUN objects in gateways transfer these payloads by using the publish service. GS_Transfer_Mode is provided with each transfer in order to indicate the service, including priority, associated with the transfer.

The GS_Transfer_Mode specify priority and discard eligibility, as defined in Clause 12 for the tunnel service.

P.14 Bulk transfer

The GIAP bulk transfer service is implemented through the bulk transfer protocol and IFO and UDO objects.

Bulk transfer is used for upload/download in half-duplex mode. An IFO acts as a client. UDOs act as servers. The UDO object identifier represents the target resource for the operation. A series of AL block transfers are controlled by the end objects to provide ordered, error-free delivery of complete blocks of a negotiated size. There is no reliance on reliable transfer in lower layers. A multi-phase transfer protocol (open, transfer and close) is employed. A series of separate requests and responses track the total transfer size. Timing attributes are defined for the UDO to assist the client in determining timeout and retry policies and to avoid congestion errors. An upload or download operation may be closed due to errors on either end.

Lease establishment for bulk transfers establishes the necessary communication resources via a communication contract prior to bulk transfer.

The G_Bulk_Open request primitive is used to initiate a bulk transfer. The target device for a bulk transfer is addressed by the GS_Network_Address, which is aIPv6Address. The target item for a bulk transfer is identified by GS_Resource, which contains the Transport_Port and the OID pointing to a specific UDO.

P.15 Alert

The GIAP alert service is implemented through the alert (alarms and events) services.

Lease establishment for alerts establishes the necessary communication resources via a communication contract to enable alert receipt. GS_Alert_Source_ID specifies Transport_Port, OID, and alert type.

P.16 Gateway configuration

There is no specific adaptation information for this item.

P.17 Device configuration

There is no specific adaptation information for this item.

Annex Q
(informative)

Exemplary GIAP adaptations for IEC 62591

NOTE The following information was derived by analysis of IEC 62591 and may contain errors. See the actual IEC standard for a full and correct understanding.

Q.1 General

Q.1.1 Overview

This standard does not define functionality for a complete gateway. It does include supporting examples that allow gateway construction by the addition of a protocol translator and a hardware interface and stack for a foreign network. Such an addition requires a separate effort to define the protocol translator.

Annex Q describes an exemplary gateway interface, called the GIAP, which is intended to be an abstraction of an underlying wireless system. In particular, it is intended to provide an abstraction for the wireless system described in this standard, and also for the wireless system described in IEC 62591.

Annex Q describes one way to implement the informative GIAP by using the IEC 62591 command set. It is not a complete design, but a reference to aid understanding.

Specific gateways (for specific fieldbuses) could include Annex Q, thus making it normative. They might also adopt a different approach.

Q.1.2 Reference

Annex Q references IEC 62591, IEC 61158-5-20, IEC 61158-6-20, and HCF_SPEC-183, which specify some of the HART commands and field encodings used by IEC 62591.

Q.1.3 Addressing

IEC 62591 device addressing and identification information includes:

- Nickname: a 2-octet short identifier for a device;
- Unique ID: an 8-octet globally unique identifier formed by HCF OUI = 0x00 1B1E + 5-octet HART Unique ID, together conforming to EUI64Address requirements;
- Long Tag: a 32-octet human-readable string.

The GIAP interface uses logical IPv6Addresses. Most IEC 62591 commands use nicknames. IEC 62591 gateways are required to implement command 841 (read network device identity using nickname) using a nickname that returns a unique ID and a long tag for a nickname. Command 832 (read network device identity using Unique ID) converts the unique ID to the nickname and long tag of a device.

It is recommended to map the unique ID into the low octets of the longer GIAP address.

Q.1.4 Stack interface

IEC 62591 describes its highest interface as an interface to the NL. The NL interface description receives parameters that it uses to invoke a TL. Regardless of the interface description, the over-the-air packet encapsulates the TL header within an NL payload.

The TL payload encapsulates one or more HART or IEC 62591 commands, both requests and responses. Annex Q describes the mapping of the GIAP services to commands that are carried by the TL.

Q.1.5 Tunneling

IEC 62591 gateways are required to tunnel HART commands. This means that a gateway includes a foreign network (the host interface) connected to the gateway and the gateway will tunnel HART commands through the foreign network.

Q.1.6 Entities

The virtual gateway, network manager, host interface (host applications) and network interface (network devices) are all IEC 62591 entities that implement (issue and respond to) HART and IEC 62591 commands. The network manager has exclusive communication to a security manager. All communication between the network manager and the network devices and all communication between the host applications and the network devices is routed through the virtual gateway, which acts as a command routing hub. The virtual gateway itself also implements certain commands. The virtual gateway communicates to the network devices through one or more network access points as well as interposing network devices that perform routing.

Q.1.7 Delayed response

HART incorporates a delayed response mechanism, where a first response indicates that the command was received but that the actual response is delayed due to extended processing requirements. The GIAP services require handling of delayed responses within the gateway. An error is returned if a command that expects an acknowledgment is not acknowledged.

Q.2 Parameters

GS_Network_Address is a logical IPv6Address used to identify a specific IEC 62591 device within a network.

GS_Unique_Device_ID is a device-unique identifier in EUI64Address format, used to identify a unique IEC 62591 device. All gateways share a unique ID of 0xF9 8100 0002.

GS_Network_ID indicates an IEC 62591 network that is accessible through the gateway. IEC 62591 defines a 16-bit ID. IEC 62591 specifies a single gateway per network. A multi-mode gateway specifies multiple networks per gateway and uses the network ID to identify the specific network associated with an IEC 62591 virtual gateway.

Q.3 Session

Multiple sessions may be established through a gateway. Each session is used to communicate with a specific network as indicated by the GS_Network_ID that is provided when the session is invoked.

IEC 62591 includes a different concept that is also called a session. This session refers to an end-to-end security session. Annex Q does not refer to the security session, but the GIAP session.

The session service releases IEC 62591 virtual gateway resources when a session ends explicitly or by timer expiration by using the following commands:

- release all leases;
- release unused communication resources;

- release unused cache.

Q.4 Lease

A lease is used to allocate and release specific communication resources within the context of a session.

NOTE IEC 62591 “services” are allocated communication path resources from a requesting device (including the gateway) to a destination. Services are requested from the network manager and identified by a service ID. Services have independent bandwidth and latency guarantees, based on service allocation requests. The network manager handles establishment and management of intermediate resources, such as common (shared) routes, based on requests.

A lease is established with command 799 (request service). This command is used to request from the network manager a connection to another device (a service) with specified bandwidth and latency.

The service is identified by a service ID (maps to GS_Lease_ID).

GS_Lease_Period is set by the protocol translator.

GS_Lease_Type is defined by the service request flags and the service application domain.

GS_Protocol_Type is defined in Annex M.

The nickname specifies the address of the gateway peer for the service (maps to GS_Network_Address_List which includes a single GS_Network_Address). IEC 62591 includes multicast mechanisms, but not for services. Device level peer-to-peer is possible within the protocol, but not recommended due to security concerns.

GS_Resource is unused in this context, so is set to 0.

The period/latency maps to GS_Lease_Parameters (GS_Period, GS_Phase, and GS_Stale_Limit).

Command 801 (delete service) is used to notify a device of the deletion of a specific service (based on the service ID) due to peer request or network manager decision.

Q.5 Device list report

An IEC 62591 gateway is required to implement command 814 (read device list entities). This command retrieves a list of the unique IDs for the devices known to the gateway.

All devices returned are on the active device list. Whitelist and blacklist indication are maintained in the network manager and within the gateway.

GS_Network_Address, GS_Unique_Device_ID, GS_Manufacturer, GS_Model, and GS_Revision are returned for each device.

Q.6 Topology report

The topology report returns a list of devices (GS_Device_List), their address (GS_Network_Address), and related information. The device list report identifies the devices in a system.

An IEC 62591 gateway is required to implement command 834 (read network topology information). This command is used to retrieve the graph information (GS_Graph_List) for a specific device. Retrieved information includes a list of Graph IDs (GS_Graph_ID) for the graphs that the device participates in and a list of nicknames for the neighbors in the graph (associated to GS_Network_Address).

An IEC 62591 gateway is required to implement command 833 (read network device's neighbor health), which returns the set of neighbors of a specific device. Each element in the list returns the neighbor nickname (which maps to GS_Network_Address within GS_Neighbor_List).

Q.7 Schedule report

The schedule report service returns schedule information for a specific device identified by GS_Network_Address. The device list report may be used to identify the devices in the system.

Command 783 (read superframe list, normally used by the network manager) is used to retrieve the list of superframes and their related information from a specific device. Retrieved information includes the superframe ID (GS_Superframe_ID), the number of slots (GS_Num_Time_Slots) and superframe mode flags (HCF_SPEC-183:2013, Table 47).

GS_Slot_Size is fixed to 10 ms. GS_Start_Time is calculated from SuperframeSlot = (Absolute Slot Number) % Superframe.NumSlots.

Command 784 (read link list, normally used by the network manager) is used to retrieve information about the link entries from a specific device. Link entries are related to slot usage within superframes. Retrieved information includes the Superframe ID (GS_Superframe_ID), the slot number in the superframe, the channel (GS_Channel), linkOptions (HCF_SPEC-183:2013, Table 46), linkType (HCF_SPEC-183:2013, Table 45), and nickname (associated to GS_Network_Address) of the link neighbor to build GS_Link_List.

GS_Channel_List contains a list of whitelist and blacklist channels as defined by GS_Channel_Status to reach GS_Channel_Number. GS_Channel_Number maps to Index = 0 for IEEE 802.15.4 channel = 11, 2,405 MHz ... Index = 14 for IEEE 802.15.4 channel = 25, 2,475 MHz. Command 817 (read channel blacklist) is used to identify the GS_Channel_Status for each channel.

Q.8 Device health report

The device health report returns device health information for a list of devices (GS_Device_List) each identified by GS_Network_Address.

All IEC 62591 devices implement and periodically publish command 779 (report device neighbor health) to make information available to the network manager and applications.

An IEC 62591 gateway is required to implement command 840 (read network device's statistics), which reports most of the command 779 information (no power status). This command uses a Unique ID to retrieve a variety of information related to a specific device, including:

- number of DPDUs generated by this device (GS_DPDUs_Transmitted);
- number of DPDUs terminated by this device (GS_DPDUs_Failed_Transmission);
- number of DL MIC failures (GS_DPDUs_Received, GS_DPDUs_Failed_Reception);
- number of NL MIC failures (GS_DPDUs_Received, GS_DPDUs_Failed_Reception);

- number of CRC errors (GS_DPDUs_Received, GS_DPDUs_Failed_Reception).

Command 840 is used multiple times to gather information for each device in the list.

Q.9 Neighbor health report

Neighbor health is periodically published to the network manager by command 780 (report neighbor health list). Neighbor signal strength is periodically published to the network manager by command 787 (report neighbor signal levels), which duplicates information in command 780.

G_Neighbor_Health_Report returns a list of link-level connection quality information for the set of neighbors of a specific device. The service is primarily implemented by command 833.

A list of devices known to the gateway (and each device address GS_Network_Address) may be retrieved by using the GIAP device list report service (G_Device_List_Report).

An IEC 62591 gateway is required to implement command 833 (read network device's neighbor health) which returns a list of link-level connection quality information for the set of neighbors of a specific device. Each element in the list returns the neighbor nickname (which maps to GS_Network_Address), the receive signal level in dB (GS_Signal_Strength), the number of packets transmitted to the neighbor (GS_DPDUs_Transmitted), the number of failed transmissions to the neighbor where no ACK/NAK DPDU was received (GS_DPDUs_Failed_Transmission), and the packets received from neighbor (GS_DPDUs_Received).

GS_Link_Status = 1 indicates that the neighbor is available for communication. GS_Link_Status = 0 indicates that the neighbor is unavailable for communication.

An IEC 62591 gateway is required to implement command 840 (read network device's statistics), which reports GS_DPDUs_Failed_Reception as described in the device health report clause.

GS_Signal_Quality is not available, so is set to the maximum quality value.

Q.10 Network health report

The device health report and neighbor health report are used to determine GS_Device_Health_List and GS_Network_Health.

An IEC 62591 gateway is required to implement command 840 (read network device's statistics). This command uses a Unique ID to retrieve a variety of information related to a specific device, including:

- number of joins (GS_Join_Count);
- date of most recent join and time of join (GS_Start_Date);
- average latency from the gateway to this node (GS_GPDU_Latency).

ASN is a count of all slots that have occurred since forming the network. It always increments and is never reset. ASN is 5-octets long. ASN 0 is when the network is born. GS_Start_Date and GS_Current_Date are derived from ASN.

Q.11 Time

IEC 62591 network time is measured relative to the absolute slot number 0 (ASN 0), which is the time when the network was last restarted. Time advances in 10 ms increments per slot.

Time distribution is configured by the network manager by using command 971 (write neighbor property flag) to specify a neighbor with the neighbor flags (0x01 time source, HCF_SPEC-183:2013, Table 59) indicating a specific neighbor as a time source. The IEC 62591 gateway is always configured as the source of network time.

Slot time is updated through neighbors by synchronization via time errors seen in packet exchanges (i.e., an ACK/NAK DPDU's TsError field).

The virtual gateway is required to synchronize with an external time source at least once per hour. UTC time is mapped to slot time from an external reference through the gateway. The mapping of ASN 0 to UTC is broadcast from the gateway. Command 793 (write UTC time mapping) is a gateway command that allows the network manager to set the mapping of the start of ASN 0 to UTC time on a device.

GS_Time is based on TAI time. UTC time is based on TAI time with leap seconds added at irregular intervals. This service applies time updates through the GIAP. TAI and UTC time updates occur due to drift. UTC adds additional updates due to leap seconds. A conversion is necessary to the internal HART time format from and to GS_Time: HART date 3 octets, time of day, 3 octets.

Command 794 (read UTC time mapping) is a gateway command that allows a device or the network manager to set and read the mapping of the start of ASN 0 to UTC time. GS_Command is used to set and read GS_Time within the gateway for synchronization purposes. Command 89 (set real-time clock) is used to set the time. Command 90 (read real-time clock) is used to read the current time.

Q.12 Client/server

Unless specified elsewhere in Annex Q, the gateway tunnels all HART commands through the GIAP client/server service. These commands are issued from a master to a slave (field device). The master assumes the client role and the slave assumes the server role.

The commands follow a request/response format. Request data octets are sent from the client to the server in GS_Request_Data. Response data octets are returned from the server to the client in GS_Response_Data. The command-specific response codes are mapped into GS_Status.

The GS_Buffer flag is set or cleared to indicate whether a command is to be buffered. The following commands are buffered:

- 0: read unique id
- 11: read unique id associated with tag
- 13: read tag, descriptor, date
- 20: read long tag
- 21: read unique id associated with long tag
- 48: read additional status
- 50: read dynamic variable assignments
- 18: write tag, descriptor, date
- 22: write long tag

- 25: write primary variable range values
- 44: write primary variable units

Multiple server responses may be received with the same GS_Transaction_ID in the case of a delayed response.

Client/server priority is established via the GS_Transfer_Mode.

IEC 62591 priority falls into one of four levels, command (highest priority), process data, normal, and alarm (lowest priority). Command priority is reserved for packets containing network control, configuration and diagnostics. Process-data priority packets contain process data and are refused when three-quarters of a device's packet buffers are full. Alarm priority packets contain alarms and events. Only a single alarm priority packet is buffered. Normal priority packets are all other packets and are refused when one-half of a device's packet buffers are full.

GS_Transaction_Info is not required.

Q.13 Publish/subscribe

Q.13.1 General

The GIAP publish/subscribe service is implemented through publication of commands by the IEC 62591 devices using burst mode. Adapters are able to publish on behalf of non-native sub-devices. IEC 62591 natively aggregates published commands where the time aligns and command 78 (read aggregated commands) is not required.

Normally, a gateway subscribes to a device publication. Within G_Subscribe, GS_Publish_Data returns the published data.

It is required that a lease be acquired for the subscription (obtain GS_Lease_ID). Lease establishment allocates resources between the gateway and the device using command 799.

The G_Publish_Watchdog indication is received if the publication is not received by the GS_Stale_Limit.

Q.13.2 Lease establishment

A subscription lease is established through the lease service. GS_Resource specifies the subscription information (command number and process variable list) to the lease service.

Command 108 (write publish data mode command number) is used to select the command to be published.

If command 108 specifies universal command 9 (read device variables) or common practice command 33 (read device variables), process variables will be assigned to slots for publication. Command 107 (write publish data device variables) is used to assign the slots.

Command 103 (write publish data period) selects the minimum (GS_Period, GS_Phase) and maximum update period (GS_Stale_Limit) for a publication (in 1/32 ms increments up to 3 600 s; requested and actual values may differ).

Command 104 (write publish data trigger) sets a trigger condition (GS_Update_Policy) for publication (continuous/windowed/rising and a level) resulting in dynamic changes to publication time. Publication occurs at least as often as when the maximum period is reached.

Command 109 (publish data mode control) turns publishing on and off. The publication source device contacts the network manager to request bandwidth.

Q.13.3 Buffering

The following commands are buffered:

- 1: read primary variable
- 2: read current & percent
- 3: read all variables
- 9: device variables and status
- 33: read device variables
- 123: read trend
- Device-specific

Q.14 Bulk transfer

The GIAP bulk transfer service corresponds to the AL provided block transfer. Operation permits upload/download (GS_Mode) in either half or full duplex modes, and relies on the TL to provide a series of application level block transfers. The transport segments and reassembles based on limited MTU in lower layers and provides error free delivery of complete blocks (all pieces are in order).

The operation uses several phases, including open (G_Bulk_Open), transfer (G_Bulk_Transfer), reset, and close (G_Bulk_Close). New commands were created to execute these phases. A master opens a session (command 111) with a slave to initiate the operation (GS_Transfer_ID links the phases of this operation). The master proposes the block sizes (GS_Block_Size), and the slave may reduce the size. A port (an octet) identifies the target resource (GS_Resource) for the operation (firmware, parameters, and log file). The total size is not stated (GS_Item_Size = 0) and may not be known even to the application (such as a continuous stream of samples organized in blocks). There is an octet counter selected by each end to track progress. Command 112 is organized such that the request contains download data (GS_Bulk_Data) and the response has upload data (GS_Bulk_Data). The request creates an indication in the slave; the response contains an indication in the master. The session is closed on errors. No rule exists on how to deal with the partial data set. The delayed response mechanism is mentioned in status, but is not described further.

Q.15 Alert

The GIAP alert service is implemented through several mechanisms. Locally buffered changes include burst mode updates (process changes), event notification (general alarms and events), device status changes, device configuration changes, network topology changes, and network schedule changes.

Change notification simply indicates a change, and further action is required to retrieve altered information from the gateway buffers. The gateway entity acknowledges event arrival to devices. Publications and alerts are stored in the gateway entity. The gateway entity acknowledges alert arrival to devices.

For example, the gateway often internally uses HART command 115..118 to set up change notification and HART command 119 to indicate that changes have occurred.

Events are configured with assigned event numbers on a per-device basis.

Command 116 (write event notification bit mask) configures the event mask that is used to trigger an event notification for a specific event. The event mask corresponds to command 48 (read additional device status), which refers to common tables 14, 17, 29, 30, 27, 31, 32, 28 and device specific status.

Command 117 controls the timing of event notifications. Event notification uses burst mode for delivery when an event is triggered. A de-bounce period is specified to prevent events that are too short from triggering a burst message. A retry time (desired burst period) and a maximum update time (maximum burst period) set the burst transfer timing if an event triggers a message.

Command 118 (event notification control) is used to enable or disable an event notification for a specific event.

Command 119 (acknowledge event notification) is used to acknowledge the event notification and clear the event from being sent in the burst updates. Other events may be in queue.

Command 115 (read event notification summary) is used to determine the configuration of an event based on a specific event number.

The following commands are buffered:

- 119 read event notification status (time stamp + device status + command 48).
- Command 788 (alarm path down), command 789 (alarm source route failed), command 790 (alarm graph route failed), and command 791 (alarm TL failed) report communication failures to the network manager.

IEC 62591 gateway command 836 (write update notification bit mask for a device) registers a client for notification updates. The device is addressed by the unique ID and given a set of change notification flags. Codes exist for BurstMode, EventNotification, DeviceStatus, DeviceConfiguration, NetworkTopology (gateway or NM), and NetworkSchedule (gateway or NM). This is used in G_Alert_Subscription to subscribe (by providing a GS_Subscription_List with GS_Alert_Source ID, GS_Subscribe, and GS_Enable for a specific device GS_Network_Address and a specific category GS_Category).

IEC 62591 gateway command 838 (read update notification bit mask for a device) returns a list of the update notifications for a device. This is used in G_Alert_Subscription to identify the subscriptions.

IEC 62591 gateway command 839 (change notification) is sent by the gateway to a client and returns a list of up to 10 change notifications (cached commands) for a device. Each change results in a single G_Alert_Notification.

Q.16 Gateway configuration

There is no specific adaptation information for this item.

Q.17 Device configuration

There is no specific adaptation information for this item.

Annex R
(informative)

Host system interface to standard-compliant devices via a gateway

R.1 Background

R.1.1 Host system integration reference model

A simplified reference model for a standard-compliant device/host system integration is depicted in Figure R.1.

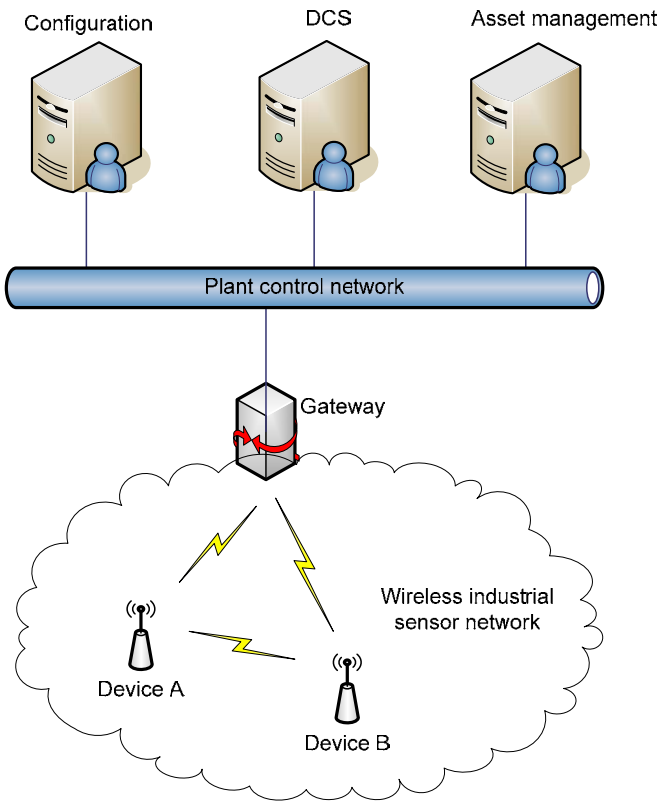


Figure R.1 – Host integration reference model

R.1.2 Asset management tools

Asset management involves overseeing the health of the system's assets by monitoring health related conditions in order to identify a potential problem before the process or plant operation is affected. Host systems provide an asset management tool or set of tools to fulfill the asset management function, with goals of lowering maintenance costs, reducing down-time, and ensuring that appropriate product quality levels are met.

R.1.3 Configuration tools

Once the system design has been established, and the system components identified, the operation of the components in the overall system needs to be configured. Host systems provide a configuration tool or set of tools that support system component configuration and define component operation in the system.

R.1.4 Distributed control system

A distributed control system (DCS) is a control system that supports a process wherein the control elements are geographically distributed. These distributed elements are connected by communication networks, which are used for communicating with the distributed elements.

R.1.5 Gateway

A gateway connects the host systems with the network. See Annex U for more information regarding the gateway.

R.2 Device application data integration with host systems

R.2.1 General

There are two generic means for host systems to integrate application data from connected devices:

- integration via protocol mapping; and
- integration via protocol tunneling.

R.2.2 Native protocol integration via mapping

Existing host systems may integrate device application data by mapping the relationship between the devices and data to the information handling performed by the existing host system. This mapping function is usually performed by a gateway between the existing host system and the wireless industrial sensor network (WISN).

R.2.3 Legacy device protocol integration via tunneling

Existing host systems may integrate application data from existing legacy devices that are using the WISN application tunneling capability in the same manner by which it presently integrates the application data from the legacy devices.

R.3 Host system configuration tool

R.3.1 General

Host systems usually support either one or both of two generic integration methods for configuring field devices:

- electronic device description language (EDDL);
- field device tool/device type manager (FDT/DTM).

R.3.2 Host configuration using electronic device description language

IEC 61804-3, which deals with EDDL, describes a generic language for describing automation device properties. EDDL can describe device functions, interactions supported by a device, device-supported objects, and other properties.

EDDL is used by a device vendor to create an electronic device definition (EDD) file that corresponds to a particular device. An EDD file is an operating system and automation system independent structured ASCII text file that describes the capabilities of a device to allow integration of the device with a host DCS system. This independence enables vendors to describe their devices in a manner that enables vendor independent interworkability and constrained interoperability of the device across host systems. EDD files describe device data, device vendor desired user interface characteristics, and device command handling, such as command ordering and timing.

Host DCSs provide tools to interpret EDD files in order to configure and handle the device, such as for monitoring or parameter handling, to support control applications.

EDDs are defined by device vendors and tested by the appropriate fieldbus supporting organization.

Figure R.2 represents configuration using a DD file.

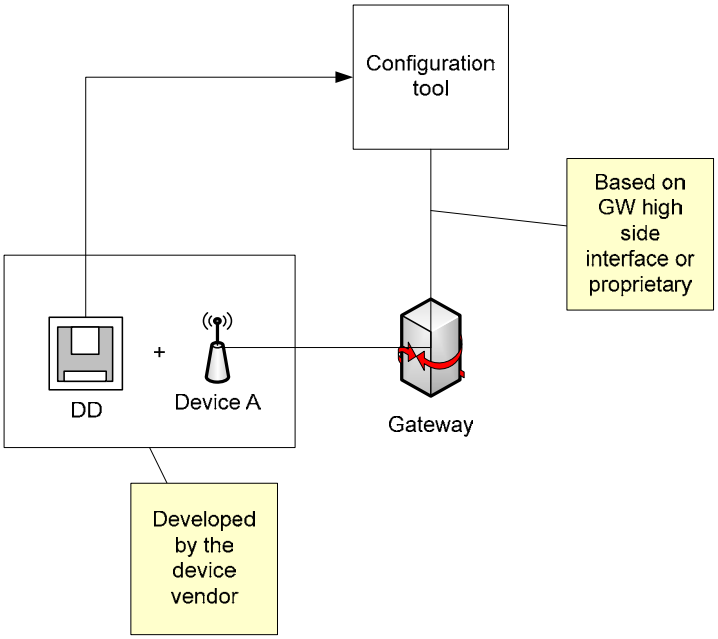


Figure R.2 – Configuration using an electronic device definition

R.3.3 Host configuration using field device tool/device type manager

The device functionality described by EDD is limited by IEC 61804-3. Additional device functionality (if any) that cannot be described via EDD can be supported via proprietary plug-ins or snap-ons. To provide this greater support, field device tool/device type manager (FDT/DTM) technology may be used. FDT/DTM technology requires, for example, FDT PDU application support in the DCS. For further information on FDT/DTM, consult the FDT Group.

Figure R.3 represents a configuration using the FDT/DTM approach.

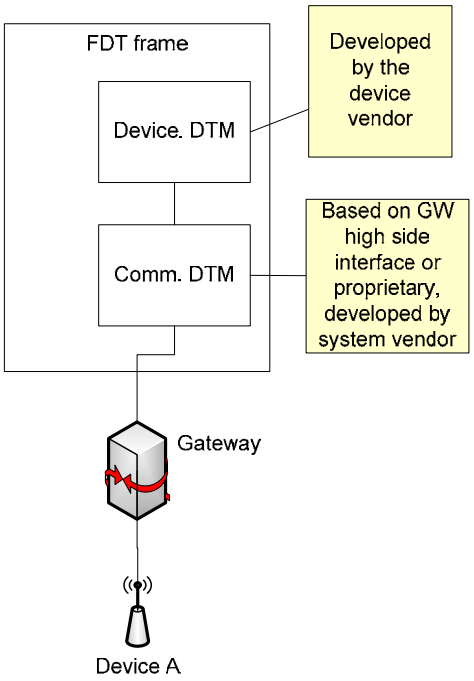


Figure R.3 – Configuration using FDT/DTM approach

R.4 Field device/distributed control systems integration

R.4.1 General

Distributed control systems usually consist of devices such as controllers, human-machine-interface (HMI) stations, data historian servers, advanced applications, etc. HMI stations, historian servers, and advanced applications often employ interfaces with rich data semantics, such as OPC. Communication with controllers usually employs simpler protocols, such as Modbus, or Foundation Fieldbus High Speed Ethernet (FF-HSE).

R.4.2 Foundation Fieldbus High Speed Ethernet

Application data integration with FF-HSE can, for example, be accomplished by mapping the native application data to FF transducer blocks. Application objects map to FF blocks, while object attributes map directly to the FF block parameters.

R.4.3 Modbus

Application data can integrate with Modbus by assigning a Modbus address to the gateway. The gateway then may present a set of register tables to Modbus masters. Each object attribute may be mapped to a specific register. The host system may provide automated support for the mapping, or mapping may be performed manually by the user.

R.4.4 Open connectivity for industrial automation

Open connectivity for industrial automation (OPC) allows client applications to access data in a consistent manner via an OPC server by referencing the data using a Tag.Parameter construct.

An OPC client may be supported by an OPC server in the host system or by a high-side OPC interface provided by a gateway to a standard-compliant system.

For example, this standard provides value, quality, and timestamp information in data publications, in support of OPC server access to online data. Native alarms and events also provide support for OPC client notification.

The OPC client may specify Tag.Parameter using the device name for the Tag, and a unique object name and attribute to represent the parameter (e.g., TI101.AITB1.PV). In the OPC server, the Tag is mapped to the device, the object instance maps to a particular object instance of a particular UAP, and the attribute name maps to the particular attribute identifier of the referenced object instance.

R.5 Gateway

R.5.1 General

Host system configuration of applications residing within the gateway itself, including data mapping (if necessary), is defined by the plant control network, which is the high side interface of the gateway that couples the WISN into a higher level control system. This includes, for example, configuration of a system management application or a tunneling application. Therefore, Annex R describes in generalities the type of information that needs to be configured for gateway support.

R.5.2 Devices supported

A host system configuration tool may need to establish the complement of standard-compliant devices with which the gateway will communicate.

R.5.3 Data subscription

A host system configuration tool may need to establish the configuration of the dispersion objects in the gateway for the data the gateway will receive via publication.

R.5.4 Data publication

A host system configuration tool may need to establish the configuration of the concentrator objects in the gateway for the data the gateway will itself publish.

R.5.5 Client/server access

Non-management related client/server communications may, for example, be established by the gateway on an as-needed basis through interface objects.

R.5.6 Alerts reception

A host system configuration tool may need to establish the alert categories associated with gateway-resident alert-receiving object(s) (AROs).

R.6 Asset management application support

R.6.1 General

An asset management tool may access information about a device that is either stored in or accessed via the gateway by using plant control network services.

A gateway may access information directly from a field device to satisfy asset management requests. The gateway may, for example, employ client/server services to read data, to write data, or to execute a particular method on a particular object instance within the wireless device.

A gateway may act as a pass-through for asset information directly from an asset to an asset management application via a plant control network tunnel if the plant control network supports such a tunneling capability.

R.6.2 Field device tool / device type manager

A DTM may be provided by a device vendor to provide process and device information to an asset management tool. A host system supporting an FDT PDU can employ the device DTM and a communication DTM for the gateway to acquire the information necessary to manage the device via the gateway.

R.6.3 HART

A standard-compliant device may be made to appear as a HART[14] native device on a HART asset management application (ASM) in several ways:

- Manually or using automation along with either explicitly coded or data-driven conversion rules provide a HART DD source file for the device. The HART DD file can be passed through a HART tokenizer to produce binary files representing the DD content. Most HART clients use the binary format of the of the DD files.
- Standard commands may be defined in HART to integrate ASM with this standard, such as HART commands for READ_IEC62734_ATTRIBUTE, WRITE_IEC62734_ATTRIBUTE, and EXECUTE_IEC62734_METHOD.
- Mapping tables in the gateway may be employed to define attribute value mapping that differs between this standard and HART, such as for engineering unit indices.

R.6.4 OPC

Open connectivity for industrial automation (OPC) allows client applications to access data in a consistent manner via an OPC server. An OPC client may be supported by an OPC server in the host system or by a high-side OPC interface provided by a gateway to a standard-compliant system.

For example, device health information may be provided by the OPC server to an OPC client.

---

14 HART is a registered trademark of HCF. This information is given for the convenience of users of the standard and does not constitute an endorsement of the trademark holders or any of their products. Compliance to this profile does not require use of the registered trademark. Use of the trademarks requires permission of the trade name holder.

Annex S
(informative)

Symmetric-key operation test vectors

S.1 DPDU samples

S.1.1 General

[INGREDIENTS]

- TsDur: 10464 [2^-20sec]
- Data DPDU Source EUI64: 0x00 00 00 00 00 00 00 01
- Data DPDU Key: 0xC0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
- Data DPDU Sequence Number: 0x04
- TAI Time[TAINetworkTimeValue]: 0x00 01 02 03 04 05
- Channel: 0x02
- Data DPDU Headers: 0x10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28
- Data DPDU Payload: 0x30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 55 56 57 58 59 5A 5B
- ACK DPDU Source EUI64: 0x00 00 00 00 00 00 00 02
- ACK DPDU Sequence Number: 0x05
- ACK DPDU Headers: 0x10 11 12 13 14 15 16 17 18

S.1.2 DPDU with expected DMIC32

[PRE-PROCESSED MATERIAL]

- Data DPDU Nonce: 0x00 00 00 00 00 00 00 01 04 08 0C 10 14
- Data DPDU MIC: 0xBF 5A BB 7C
- ACK DPDU Nonce: 0x00 00 00 00 00 00 00 02 04 08 0C 10 15
- ACK DPDU authentication vector: 0x10 11 12 13 14 15 16 17 18 BF 5A BB 7C
- ACK DPDU MIC: 0x74 F0 41 B3

[DELIVERABLE]

- Data DPDU: 0x10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 55 56 57 58 59 5A 5B BF 5A BB 7C
- ACK DPDU: 0x10 11 12 13 14 15 16 17 18 74 F0 41 B3

S.1.3 DPDU with expected ENC-DMIC32

[PRE-PROCESSED MATERIAL]

- Data DPDU Nonce: 0x00 00 00 00 00 00 00 01 04 08 0C 10 14
- Encrypted Data DPDU Payload: 0x23 F4 C4 3F BA 9B E4 3E D8 9B FD 36 A8 76 C7 99 27 14 E0 42 94 94 DE 64 B2 6B 14 18 51 9F 8D 11 36 F4 09 17 6B D6 A6 75 07 B1 D2 90
- Data DPDU MIC: 0xD0 F6 B2 65

- ACK DPDU Nonce: 0x00 00 00 00 00 00 00 02 04 08 0C 10 15
- ACK DPDU authentication vector: 0x10 11 12 13 14 15 16 17 18 D0 F6 B2 65
- ACK DPDU MIC: 0x26 AB 87 D2

[DELIVERABLE]

- Data DPDU: 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 23 F4 C4 3F BA 9B E4 3E D8 9B FD 36 A8 76 C7 99 27 14 E0 42 94 94 DE 64 B2 6B 14 18 51 9F 8D 11 36 F4 09 17 6B D6 A6 75 07 B1 D2 90 D0 F6 B2 65
- ACK DPDU: 0x10 11 12 13 14 15 16 17 18 26 AB 87 D2

S.2 TPDU samples

S.2.1 General

[INGREDIENTS]

- TPDU time creation[TAINetworkTimeValue]: 0x00 01 02 03 04 05
- Key: 0xC0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
- Crypto Key Identifier Mode: 0x00
- Crypto Key Identifier = 0x10
- Source EUI64Address: 0x00 00 00 00 00 00 00 01
- Source IPv6Address: 0xFE 80 00 00 00 00 00 00 00 00 00 00 00 00 00 01
- Dest IPv6Address: 0xFE 80 00 00 00 00 00 00 00 00 00 00 00 00 00 02
- Source Port: 0x00 01
- Dest port: 0x00 02
- TSDU (Application Layer Payload): 0x10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B

S.2.2 TPDU with expected ENC-TMIC-32:

[PRE-PROCESSED MATERIAL]

- TPDU Pseudo header: 0xFE 80 00 00 00 00 00 00 00 00 00 00 00 00 00 01 FE 80 00 00 00 00 00 00 00 00 00 00 00 00 00 02 00 2B 00 11 00 01 00 02
- TPDU Nonce: 0x00 00 00 00 00 00 00 01 04 08 0C 10 FF
- TPDU Security header: 0xA0 0C 10

[DELIVERABLE]

- TPDU: 0x 00 01 00 02 00 23 00 00 A0 0C 10 8E 7C 0B B9 8B CD 15 7E 59 CE 71 18 14 B7 05 FE C2 6A F1 C3 9D 05 B9 FD E6 5F 16 C9 DE 37 DE BE

S.2.3 TPDU with expected TMIC-32:

[PRE-PROCESSED MATERIAL]

- TPDU Pseudo header: 0xFE 80 00 00 00 00 00 00 00 00 00 00 00 00 00 01 FE 80 00 00 00 00 00 00 00 00 00 00 00 00 00 02 00 2B 00 11 00 01 00 02
- TPDU Nonce: 0x00 00 00 00 00 00 00 01 04 08 0C 10 FF
- TPDU Security header: 0x20 0C 10

[DELIVERABLE]

- TPDU: 0x 00 01 00 02 00 23 00 00 20 0C 10 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B 7E 8C 35 57

Annex T
(informative)

Data-link and network headers for join requests

T.1 Overview

Annex T illustrates the DL header and NL header for a typical join request.

T.2 MAC header (MHR)

The MAC header for join messages is shown in Table T.1. IEEE convention shows bit 0 on the right, which is the nominal order of transmission. Per IEEE 802.15.4 convention, the Sequence Number and Addressing fields of the MHR, when considered as unsigned integers, are transmitted lowest-weight octet (LSB) first.

NOTE IEEE 802.15.4 2,4 GHz DSSS actually transmits quartets of four bits simultaneously as 32-chip spread-spectrum signaling, so there is no “first” or “last” bit transmitted within the quartet. However, the lower-bit-weight quartet in an octet, when interpreted as an Unsigned8, is transmitted before the higher-bit-weight quartet of that same octet, and the lower-weight octet is transmitted before the higher-weight octet.

Table T.1 follows the convention of this standard, showing bit 7 on the left.

Table T.1 – Sample MHR for join request

<table>
<tr><th rowspan="2">Subfield</th><th rowspan="2">Number of octets</th><th colspan="8">bits</th></tr>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td rowspan="2">Frame Control</td><td rowspan="2">2</td><td>Reserved = 0</td><td>PAN ID Compress = 1 (yes)</td><td>ACK Request = 0 (no)</td><td>Frame Pending = 0 (no)</td><td>Security Enabled = 0 (no)</td><td colspan="3">Frame Type = 1 (Data)</td></tr>
<tr><td colspan="2">Source Addressing Mode= 3 (64-bit)</td><td colspan="2">Frame Version = 1</td><td colspan="2">Dest Addressing Mode=2 (16-bit)</td><td>Reserved = 0</td></tr>
<tr><td>Sequence Number</td><td>1</td><td colspan="8">(determined by DLE at time of transmission)</td></tr>
<tr><td rowspan="3">Addressing</td><td>2</td><td colspan="8">PAN ID (LSB), from advertisement</td></tr>
<tr><td>2</td><td colspan="8">Destination Address (16-bit, LSB), from advertisement</td></tr>
<tr><td>8</td><td colspan="8">Source Address (64-bit, LSB), device’s EUI64Address</td></tr>
</table>

T.3 DL header (DHR)

The DL header for join messages is shown in Table T.2. This example assumes that the advertisement does not specify slow-channel-hopping.

Table T.2 – Sample DHR for join request

Sub-header	octets	bits							
		7	6	5	4	3	2	1	0
DHDR	1	ACK/NAK DPDU needed = 1 (yes)	Signal quality in ACK/NAK DPDU = 0 (no)	Request EUI64Address = 0 (no)	Include DAUX = 0 (no)	Include slow channel hopping-offset = 0 (no)	Clock recipient = 1(yes)	DL version = 00	
DMXHR	1	Reserved=0			Key identifier mode = 1		Security level = 1 (MIC-32)		
	1	Crypto key identifier = 0: K_global							
DAUX	0	(absent by DHDR setting)							
DROUT	1	Compress = 1	Priority = 0 (irrelevant)				DlForwardLimit = 1		
	1	GraphID (Unsigned8) =0 (Single hop source routing)							
DADDR	1	DE = 0	LH = 0	ECN = 0	Reserved = 0				
	1	SrcAddr = 0 (Use EUI64Address in MHR)							
	1	DestAddr = 0 (Use DL16Address in MHR)							

T.4 NL header

The network header for join messages is shown in Table T.3.

Table T.3 – Network header for join messages

octets	bits							
	7	6	5	4	3	2	1	0
1	LOWPAN_IPHC dispatch = 011			LOWPAN_IPHC encoding (bits 8..12) = 11 101				
2	LOWPAN_IPHC encoding (bits 0..7) = 0111 0111							

Annex U
(informative)

Gateway role

U.1 General

U.1.1 Overview

The primary purpose of a gateway as described by this standard is to enable host-level applications to interact with wireless field devices. A large installed base of applications exists, including automation devices, controllers, and supervisory systems, which together use numerous legacy protocols, thus requiring protocol translation when interacting with wireless field devices. Such protocol translation may be present in the gateway and also in adapters to legacy wired field devices. Within this standard, the term adapter is used to identify devices that convert from a wired fieldbus protocol to a wireless fieldbus protocol on behalf of one or more field devices.[15] Such protocol translation generally serves to tunnel a foreign protocol across a wireless network as described in this standard, or to convert a legacy protocol to and from this standard's native format. The term native field device refers to a field device that functions exclusively through the usage of the native objects, native interfaces, and native message content as defined in this standard.[16] It is also possible to write or modify host-level applications to use the native application protocol directly, reducing or eliminating the need for protocol translation within a gateway.

NOTE The examples provided in Annex U are symmetric, potentially applicable without modification to both gateways and adapters. In practice, the specific foreign protocol features and the usage of a gateway or adapter relative to host-level applications and field devices will dictate the subset of the protocols that apply to each.

The description of the gateway role relates to the following capabilities:

- Interfacing foreign host-level applications:
  - directly to "native" field devices (i.e., ones conforming to this standard); and
  - indirectly to legacy wired field devices through legacy adapters.
- Interfacing host-level applications to multiple wireless systems, including a combination of one or more wireless systems as described in this standard and one or more foreign wireless systems, through a single (conceptual) device with a common high-side interface.

This standard provides supporting functionality for the construction of gateways. It does not provide complete details on how to construct any particular gateway. Annex U is strictly informative, since no gateways are specified. As such it provides a suggested basis for future construction of gateway specification, but is itself not one. No validation of the content of Annex U has occurred.

Annex U describes support functionality for foreign protocol translation needs, but does not describe details on how to perform any specific protocol translation or how to interface to any specific plant network.

---

[15] Usage of the term adapter is not uniform. Technically, an adapter is an interface from a CPU to a communication channel. Technically, a gateway is an interface from one communication channel to another communication channel, where protocol translation is used at one or more layers of the protocol suite. There is a precedent set in the automation industry to (incorrectly) use the term adapter to identify devices that convert from a wired fieldbus protocol to a wireless fieldbus protocol on behalf of one or more field devices. There is also a precedent in the automation industry to (correctly) use the term gateway to identify devices that convert from a wireless fieldbus protocol to a wired fieldbus protocol for attachment to a control system. This document adheres to this automation industry usage in an attempt to minimize confusion.

[16] The native tunnel object uses the native tunnel and publication services to carry foreign message content. A field device that requires foreign message content to perform its function cannot be considered a native device.

Legacy protocols were not designed to operate over wireless networks. They do not access information in a manner that conserves energy, and they often are intolerant of delayed access to quiescent devices that are conserving energy. The gateway support functionality of this standard is, in large part, intended to enable the construction of gateways that adapt legacy protocols to the requirements of low-energy-consumption wireless devices.

The gateway role includes a specialized UAP. Functionality is provided by AL objects and gateway internal operation. The objects use the interfaces of the communications protocol suite to support gateway high-side interface functions.

U.1.2 Notional gateway protocol suite diagrams for native devices and adapters

The diagram in Figure 17 depicts a notional gateway interfacing a host-level application (the example control system) to a wireless field device. In this case, the field device is a native device. Protocol translation is performed in the gateway to convert between the plant network protocol and this standard's native protocol. Routers may exist between the gateway and the field device, as depicted in Figure 17, but from the gateway's perspective their operation is transparent.

The diagram in Figure 19 depicts a notional gateway interfacing a host-level application (the example control system) to a wired I/O device through a wireless system that conforms to this standard. In this specific case, the interfaced I/O device is a legacy device, not a native wireless device, and thus an adapter is required. Protocol translation is performed in both the gateway and the adapter. The gateway and the adapter each convert between legacy protocols and the communication protocols specified by this standard.

NOTE 1 It is often possible to implement an adapter to a single legacy wired I/O device by performing a protocol translation to and from native formats, without carrying any foreign message content. To a gateway, such a combination of an adapter and a connected legacy device is indistinguishable from a native I/O device. No special gateway provisions are made for such devices. Additionally, there are no special gateway provisions to facilitate multiplexing of such an adapter to multiple legacy wired I/O devices.

As seen in Figure 19, a notional gateway and a notional adapter share a common structure. Both have an interface and a protocol suite for a foreign network. Both have an interface and a protocol suite as described in this standard. Both have protocol translators. The common structure extends even further – they may share common objects and a common high-side interface structure. For this reason, no separate role was described for an adapter.

NOTE 2 The differences between a gateway and an adapter relate mostly to the implementation. For example, certain legacy protocols only publish from the field, thus requiring support for producer functionality but not consumer functionality in the adapter. Other legacy protocols also support publishing to the field, so require both producer and consumer functionality. In another example, legacy engineering tools carried into the field and plugged into the legacy network behind the adapter sometimes need to use the same functions as if they were behind the gateway.

For a gateway and an adapter to be interworkable, they require common protocol translation. If the adapter converts to and from native format, the gateway may do the same. If the adapter tunnels a legacy protocol, the gateway may tunnel the same protocol.

U.1.3 Gateway scenarios

Common gateway scenarios are depicted in Figure U.1. This figure does not attempt to provide an exhaustive description of all variations; rather, it is included to illustrate the bounds of this standard.

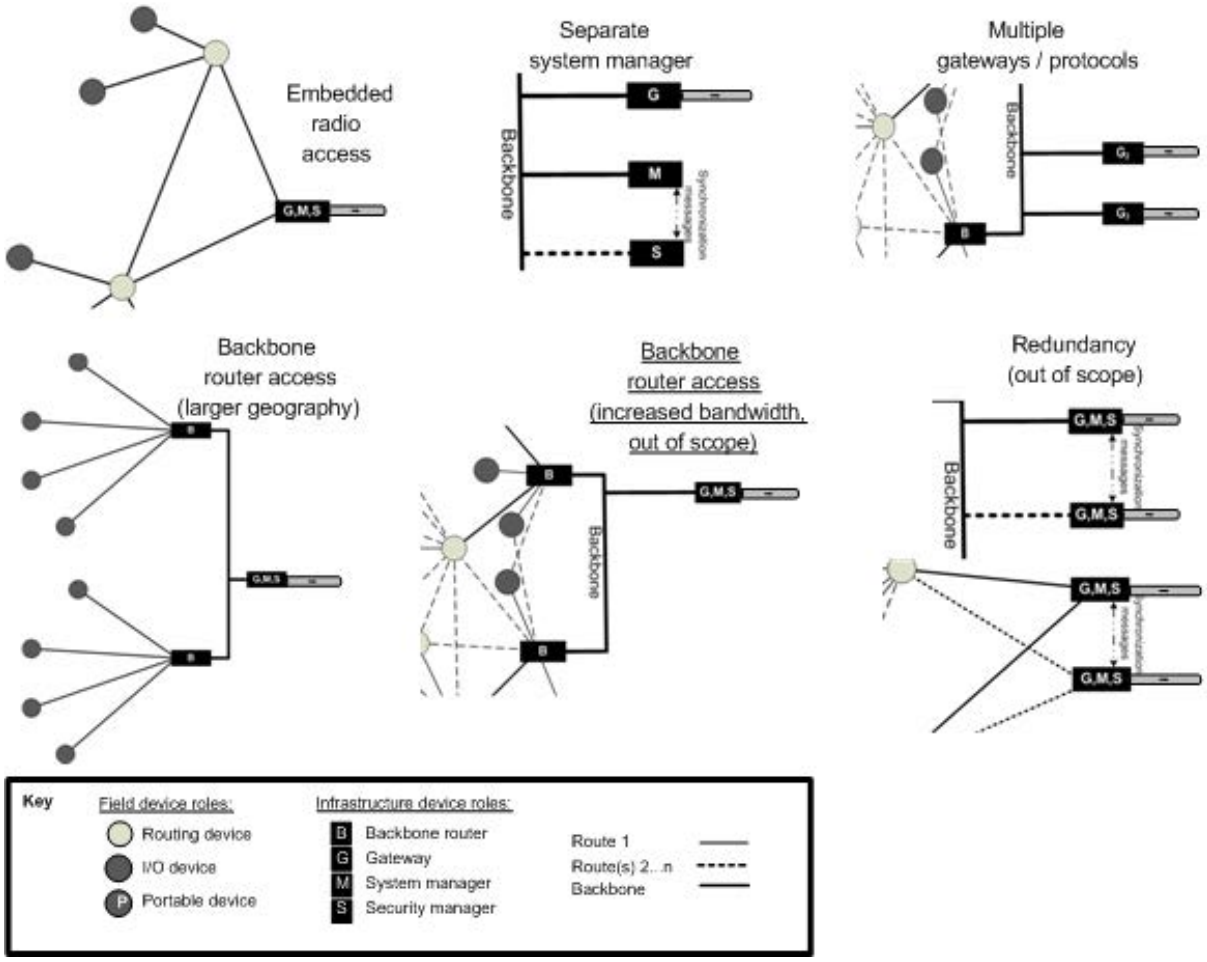


Figure U.1 – Gateway scenarios

As described in Clause 5, a gateway implements a role within the system. A variety of physical implementations are possible.

Some device implementations may have a Class A wireless interface and protocol stack embedded in the same packaging as the gateway, providing direct access to the wireless network. Independently, some device implementations may have system management and security management roles co-resident with the gateway role.

The system manager and security manager roles need not be co-resident with the gateway role. The gateway does not interact directly with the security manager, but indirectly through the system manager. The gateway functionality requires a communication path with a system manager to function as part of an operational system.

A device implementing a gateway role may use backbone routers to communicate with wireless field devices conforming to this standard.

Gateway communication with field devices through backbone routers is transparent in operation. NL extensions exist in other portions of this standard to support this transparency. It is, however, necessary to configure this routing within the gateway and the backbone routers. The backbone routers may be used to extend the geographical scope of gateway-connected devices. Backbone routers may also be used to increase bandwidth or to add redundant paths between a gateway and a mesh.

Multiple independent gateways may exist within a system. This is facilitated by independent addressing and independent communication relationships between devices. One use for

independent gateways is to support multiple independent protocols. No special provisions are made in this standard for inter-dependent operation between gateways, such as for redundancy or load sharing.

U.1.4 Basic gateway model

Gateways may follow the general model depicted in Figure U.2.

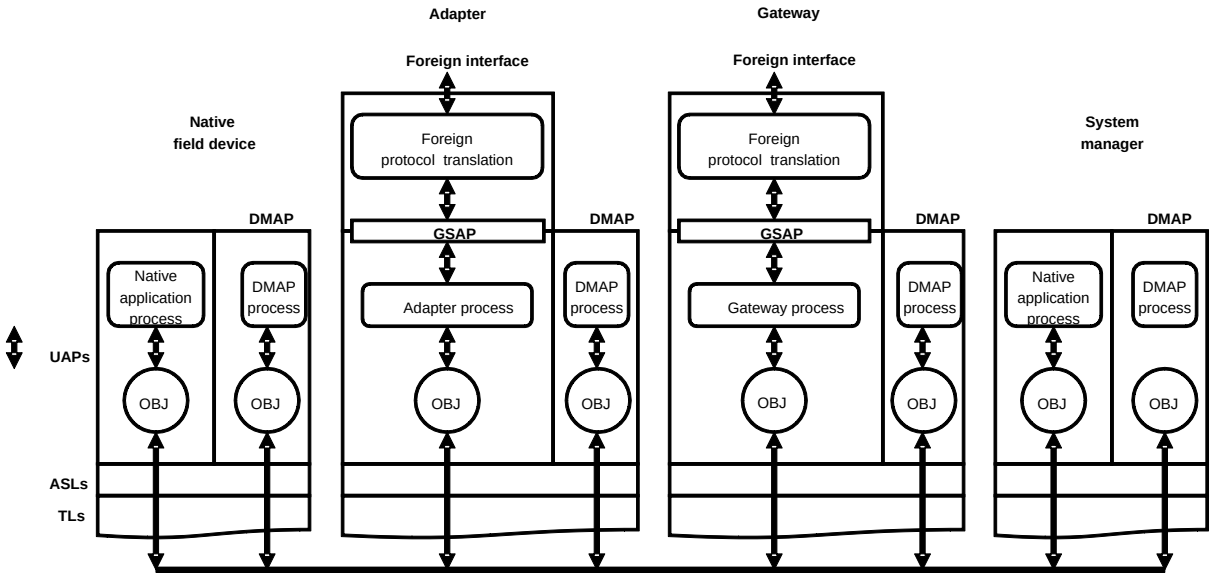


Figure U.2 – Basic gateway model

In this example, gateways and adapters host foreign protocol translators that receive and transmit foreign interface messages (usually from a control system, an asset management system, or an engineering system) and use gateway interfaces to interact with wireless devices. Gateway interfaces are provided via a high side interface that is accessed at a GIAP. Device-local protocol translators use these interfaces through the GIAP.

Protocol translation conveys application information for control, monitoring, configuration, and management. Foreign protocol messages contain this information. Tunneling, foreign protocol application communication (FPAC), and native object access methods are provided within the objects described in this standard to support protocol translation as described in Annex N. Each method entails specific tradeoffs of translation effort, energy efficiency, and performance. Practical protocol translators are likely to use a combination of these methods.

Gateways and adapters may each have application processes that interface to the protocol translators through the GIAP. Each process provides the high side interfaces by using application objects (OBJ). Inter-object communication uses the messaging methods provided through the application sublayer (ASL).

The GIAP interfaces are used for network management, protocol tunneling, upload and download, alerts, time management, and access to native application and management objects. The GIAP is described in detail in Clause U.2.

For wired automation devices, an adapter performs a symmetric function and converts foreign interface messages to an adapter high side interface (also a GIAP) via a gateway peer foreign protocol translator.

Gateway access to a native field device (from a gateway) is enabled through the same GIAP; however, the object interactions use native messaging exclusively. A native application

process interacts with the gateway or adapter process in a manner that depends on the type of the object.

Gateways, adapters, and native field devices can all be managed through the same symmetric method. The DMAP process is the peer process in this instance. Such management is specific to this standard and not necessarily to foreign protocols. Foreign protocols may provide additional device and wired fieldbus management methods that are outside the scope of this standard.

The basic gateway model includes interfacing to the system manager, which permits the system manager to be accessed via the gateway.

Backbone routers and routing devices are not shown in Figure U.2, because gateway functionality occurs within the application layer, not within the lower communication layers.

U.2 Notional GIAP

U.2.1 Summary of interfaces and primitives

The gateway portion of this standard describes a notional GIAP that can serve as a high side interface above a wireless communication protocol suite for conveying wireless information and managing wireless behavior. This notional GIAP is generic and could be used as a common interface above the AL of this standard and above other functionality in similar communication protocol suites. Annex P describes one potential implementation of the GIAP interfaces for the wireless protocol suite of this standard, using the defined AL objects and interfaces. Annex Q describes another notional GIAP interface implementation for an alternative wireless protocol suite.

NOTE 1 A primary intent of the example GIAP interfaces is to allow multimode access where a number of wireless interfaces are available to the gateway. In configurations where the path from the system manager to the plant network is only via the gateway role the GIAP supports consistent reporting information on each underlying wireless interface to promote improved coexistence. System management information can be included in a report on communication performance to identify potential interference problems, a report on topology to identify collocated devices, and a report on channel and schedule information to identify potential usage conflicts.

NOTE 2 Another intent of the example GIAP is to provide a model for the configuration and access of multiple underlying wireless networks. This potentially reduces the effort for a gateway developer if they have multiple fieldbus protocols to support. The GIAP interfaces may or may not be applicable to specialized gateway developers, such as those serving a single foreign protocol; specialized gateways may prefer to use customized gateway internal interfaces.

This notional GIAP interface is usable by a variety of protocol translators to interface to wireless communication protocol suites for conveying wireless information and managing wireless behavior. A protocol translator exists in a gateway. Depending on the implementation, a protocol translator and a GIAP may also exist in an adapter. Protocol translators will vary in complexity depending on the protocol that exists above the gateway and below the adapters. Certain protocols will use a subset of these notional GIAP interfaces. For example, a protocol may only require client/server interaction and not require publish/subscribe interfaces. Functionally, an adapter is considered a subset of a gateway and would only be expected to support a subset of these notional GIAP interfaces related to conveying wireless information.

NOTE 3 This gateway discussion does not describe protocol translation for specific fieldbus protocols.

GIAP interfaces are summarized in Table U.1.

Table U.1 – Summary of notional gateway high-side interface examples

Interface example	Interface subtype	Primitive	Description
Session	—	G_Session request G_Session confirm	A foreign protocol translator within the gateway may establish sessions on behalf of remote clients
Lease	—	G_Lease request G_Lease confirm	Leases allow the gateway to internally manage its internal communication resources on a per session basis
Device_List_Report	—	G_Device_List_Report request G_Device_List_Report confirm	Determines the devices associated with the gateway role
Topology_Report	—	G_Topology_Report request G_Topology_Report confirm	Provides a topology report related to devices in a wireless mesh. This interface may be useful if, for example, the gateway role is operating in a system in which the system management interface to the plant network is via the gateway role
Schedule_Report	—	G_Schedule_Report request G_Schedule_Report confirm	Provides detailed time slot and channel allocations on a per-device basis. This interface may be useful if, for example, the gateway role is operating in a system in which the system management interface to the plant network is via the gateway role
Device_Health_Report	—	G_Device_Health_Report request G_Device_Health_Report confirm	Device health report for devices associated with the gateway. This interface may be useful if, for example, the gateway role is operating in a system in which the system management interface to the plant network is via the gateway role
Neighbor_Health_Report	—	G_Neighbor_Health_Report request G_Neighbor_Health_Report confirm	Communication health report for the set of neighbor devices associated with a specific device that is associated with the gateway. This interface may be useful if, for example, the gateway role is operating in a system in which the system management interface to the plant network is via the gateway role

Interface example	Interface subtype	Primitive	Description
Network_Health_Report	—	G_Network_Health_Report request	Summary of communication health report for the wireless network. This interface may be useful if, for example, the gateway role is operating in a system in which the system management interface to the plant network is via the gateway role
		G_Network_Health_Report confirm	
Time	—	G_Time request	Retrieval and setting of time for the wireless network associated with the gateway
		G_Time confirm	
Client/server	—	G_Client_Server request	Provides client/server communication
		G_Client_Server indication	
		G_Client_Server response	
		G_Client_Server confirm	
Publish/subscribe	Publish	G_Publish request	Provides publish/subscribe communication
		G_Publish indication	
		G_Publish confirm	
	Subscribe	G_Subscribe request	
		G_Subscribe confirm	
	Publish_Timer	G_Publish_Timer indication	
	Subscribe_Timer	G_Subscribe_Timer indication	
	Watchdog_Timer	G_Watchdog_Timer indication	
Bulk_Transfer [a]	Open	G_Bulk_Open request	Allows upload and download of large items such as firmware images and sample buffers
		G_Bulk_Open confirm	
	Transfer	G_Bulk_Transfer request	
		G_Bulk_Transfer confirm	
	Close	G_Bulk_Close request	
		G_Bulk_Close confirm	
Alert	Subscribe	G_Alert_Subscription request	Allows subscription and receipt of specific alerts
		G_Alert_Subscription confirm	
	Notify	G_Alert_Notification indication	
Gateway_Configuration	Read	G_Read_Gateway_Configuration request	Provides read and write access to configuration attributes of the gateway
		G_Read_Gateway_Configuration confirm	
	Write	G_Write_Gateway_Configuration request	
		G_Write_Gateway_Configuration confirm	
Device_Configuration	Read	G_Read_Device_Configuration request	Allows the gateway to determine which devices are associated with it
		G_Read_Device_Configuration confirm	
	Write	G_Write_Device_Configuration request	
		G_Write_Device_Configuration confirm	

[a] The interface primitives are common to both upload and download operations.

U.2.2 Sequence of primitives

Figure U.3, Figure U.4, Figure U.5, Figure U.6, Figure U.7, Figure U.8, Figure U.9, Figure U.10, Figure U.11, Figure U.12, Figure U.13, Figure U.14, and Figure U.15 show the sequences of primitives for gateway high side interfaces. The figures are described in terms of a gateway-internal client, a gateway entity, a device client, and a device entity. A gateway-internal client is a user of the GIAP interfaces within a gateway. A gateway entity is a provider of GIAP interfaces within the gateway. The provision of the interfaces entails additional interactions across the wireless network to one or more devices. A device client is a user of GIAP interfaces within a device. A device entity is a provider of GIAP interfaces within the device.

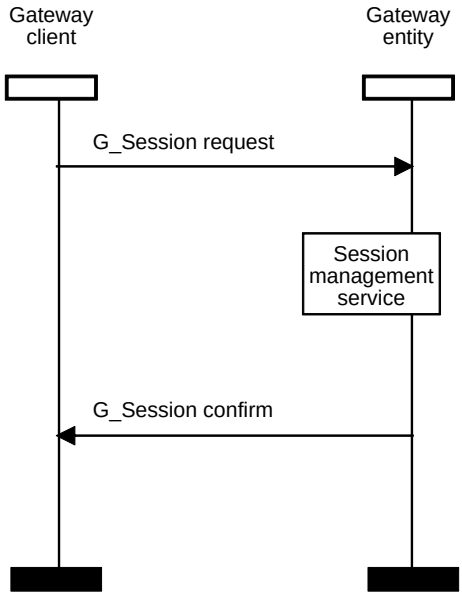


Figure U.3 – Internal sequence of primitives for session interface

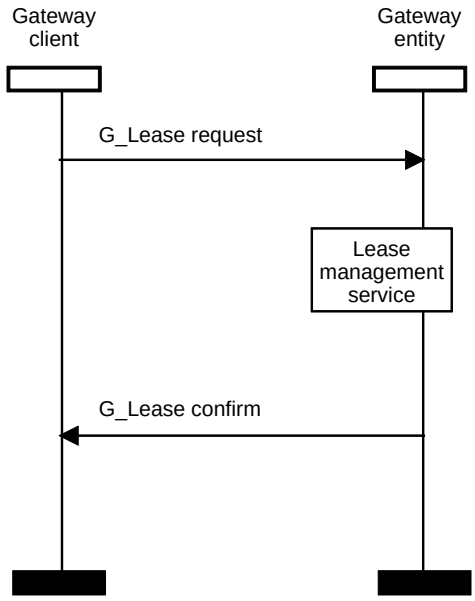


Figure U.4 – Internal sequence of primitives for lease management interface

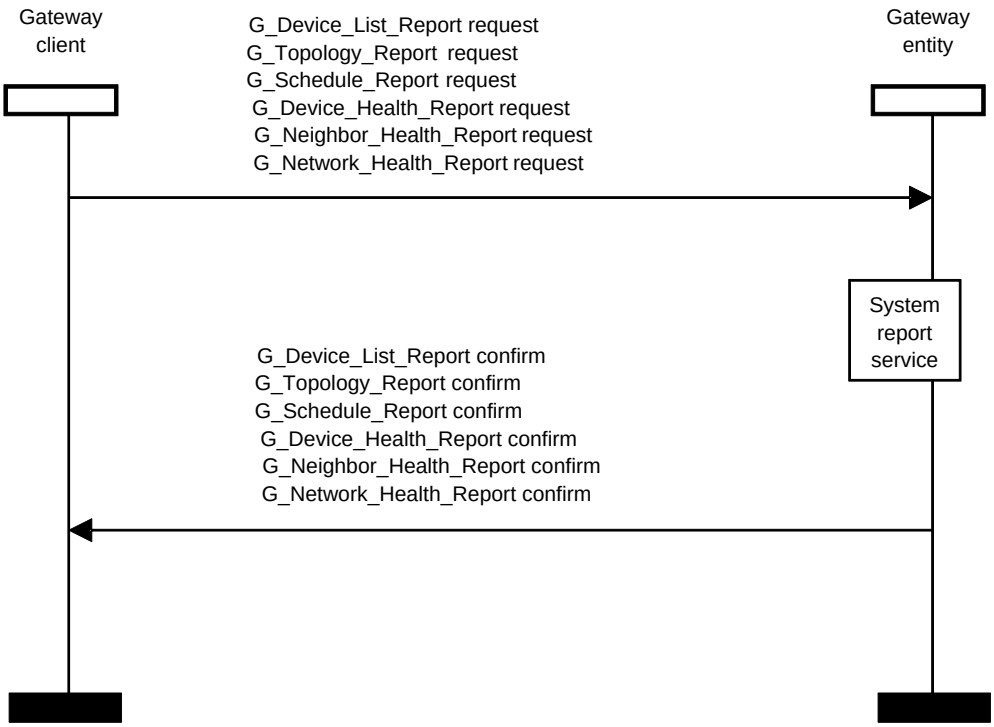


Figure U.5 – Internal sequence of primitives for system report interfaces

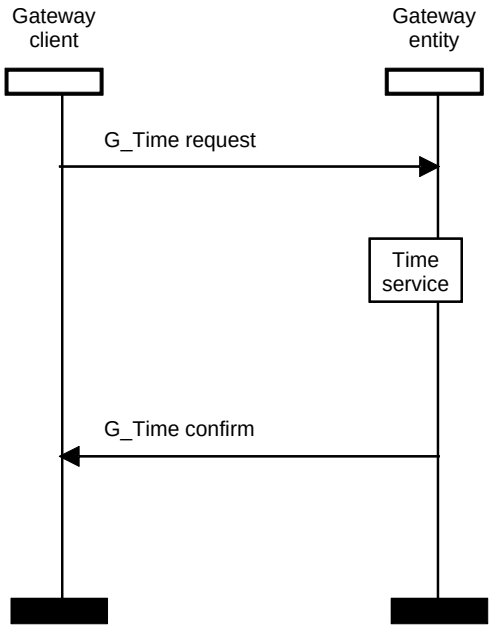


Figure U.6 – Internal sequence of primitives for time interface

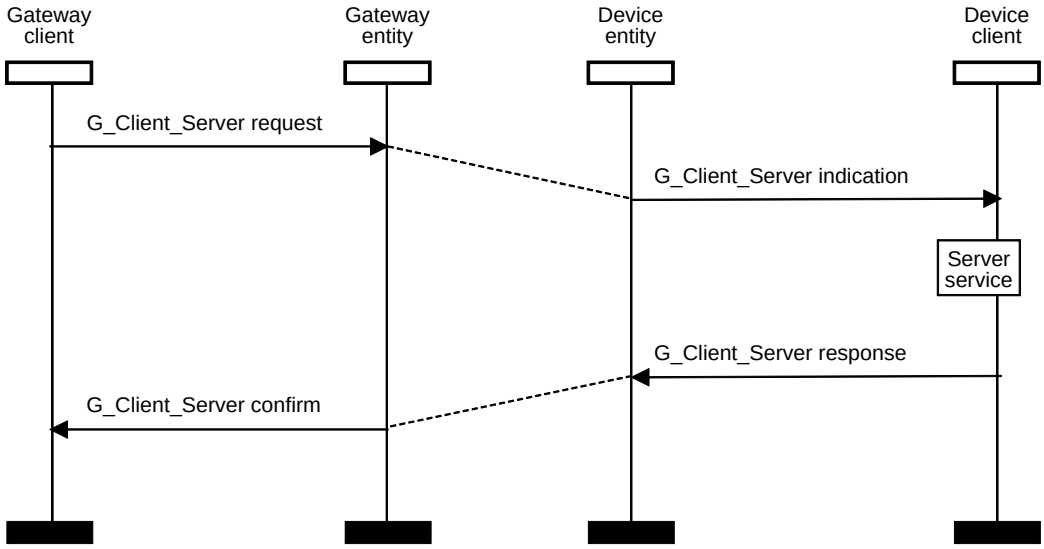


Figure U.7 – Internal sequence of primitives for client/server interface initiated from gateway to an adapter device

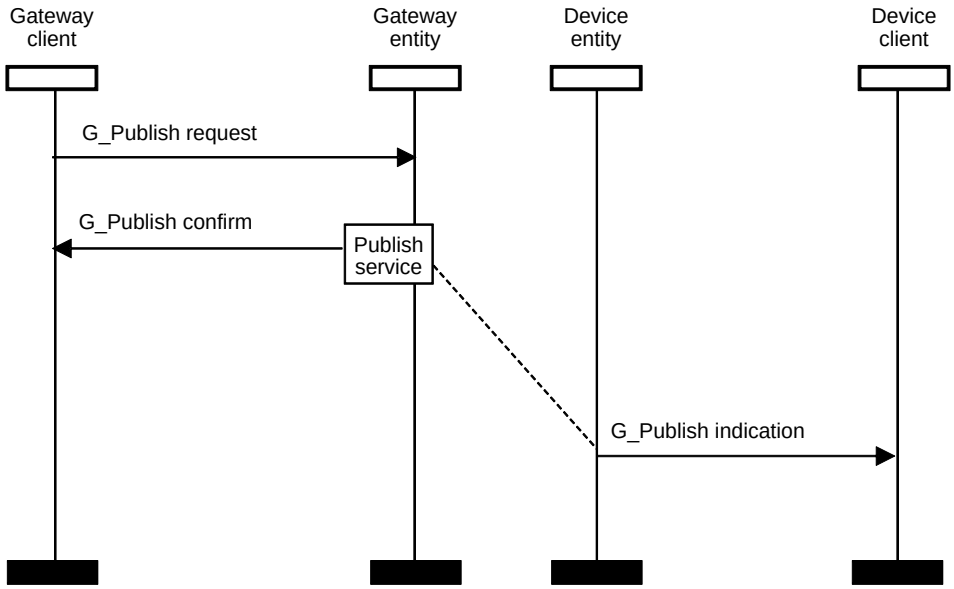


Figure U.8 – Internal sequence of primitives for publish interface initiated from gateway to an adapter device

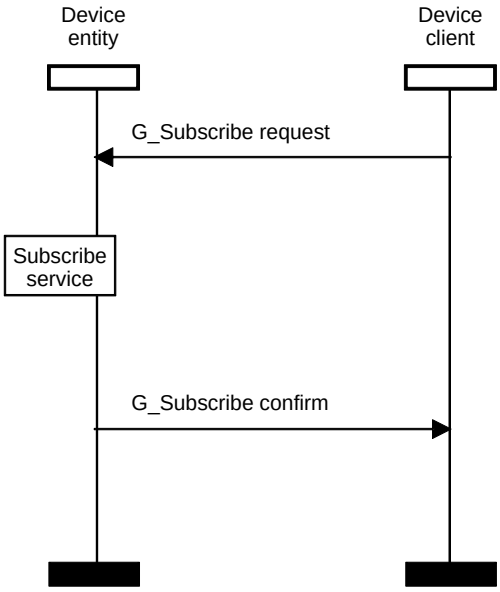


Figure U.9 – Internal sequence of primitives for subscribe interface initiated from an adapter device

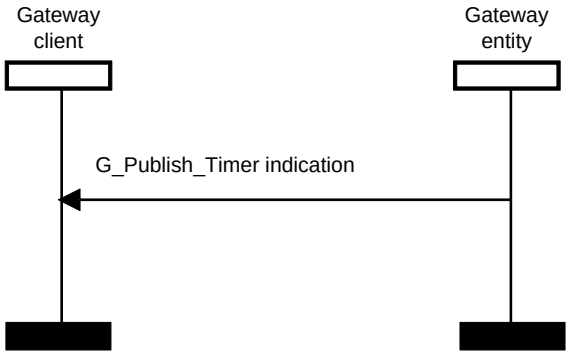


Figure U.10 – Internal sequence of primitives for publisher timer initiated from gateway to an adapter device

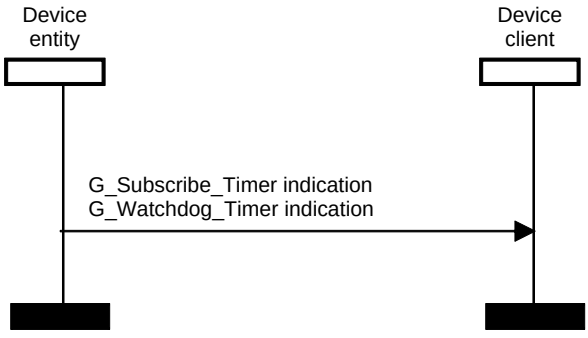


Figure U.11 – Internal sequence of primitives for subscriber timers initiated from an adapter device

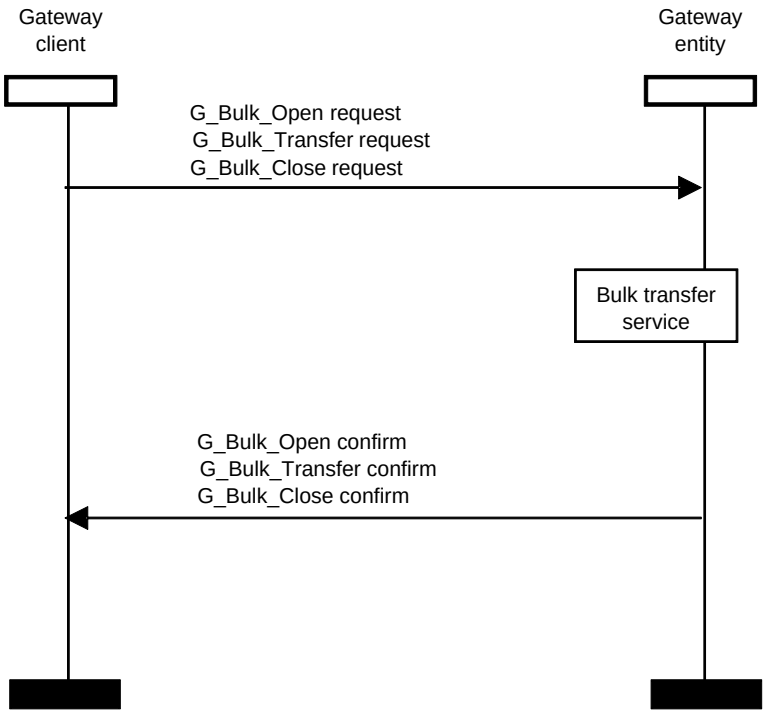


Figure U.12 – Internal sequence of primitives for the bulk transfer interface

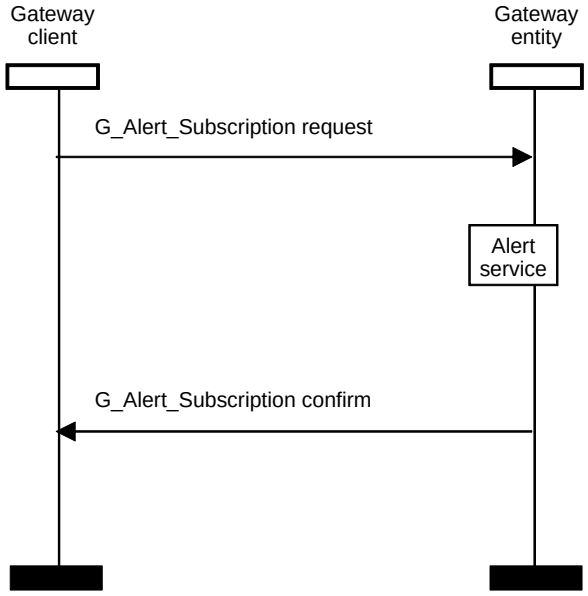


Figure U.13 – Internal sequence of primitives for the alert subscription interface

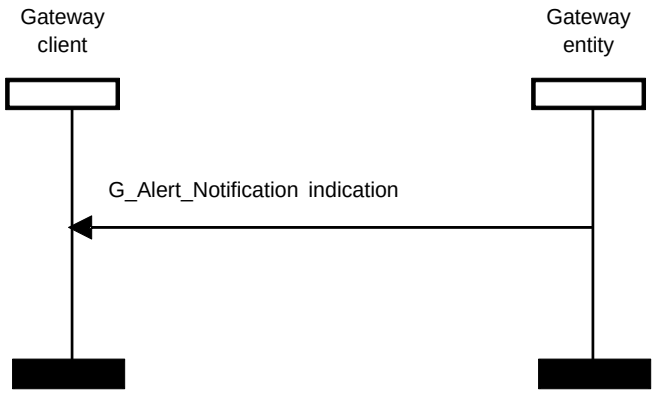


Figure U.14 – Internal sequence of primitives for the alert notification interface

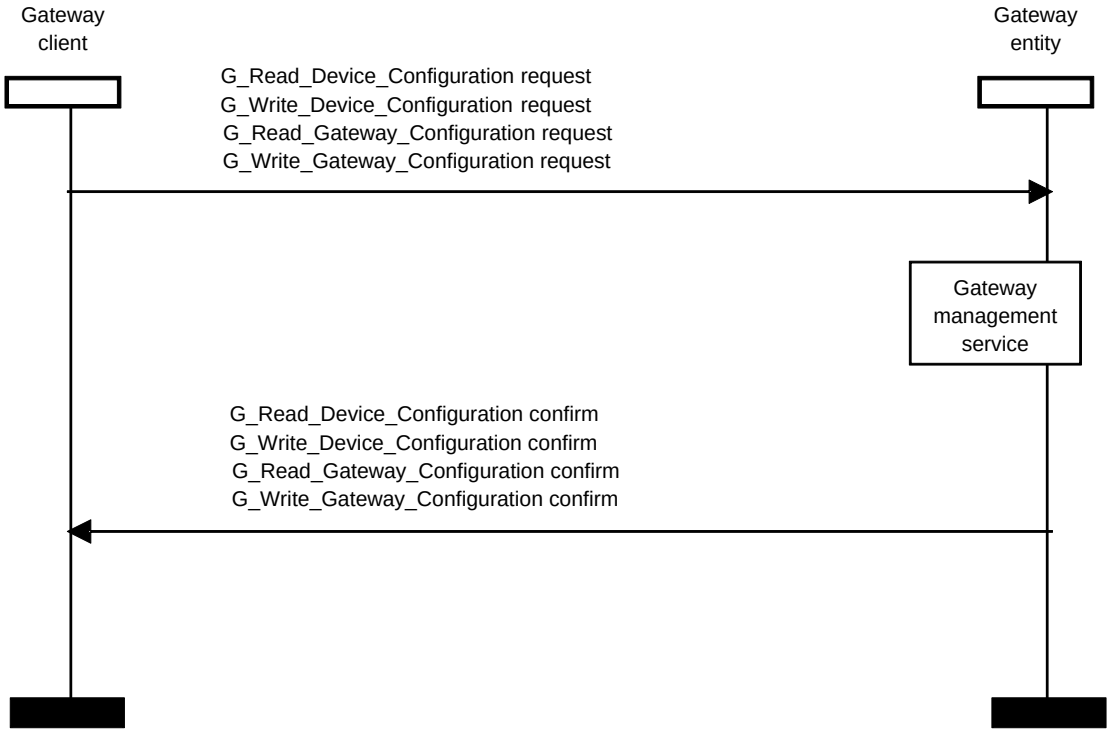


Figure U.15 – Internal sequence of primitives for gateway management interfaces

U.2.3 Detailed description of parameters

U.2.3.1 General

Parameters that are common to multiple interfaces are described in U.2.3. Parameters that are unique to an interface are described within that interface.

NOTE Since this standard does not define any gateways, and because the GIAP discussed is notional, all of these parameters are also strictly notional, as are all statements about them.

U.2.3.2 Parameter GS_Session_ID

The parameter GS_Session_ID uniquely identifies a specific session.

A valid session identifier that has not expired is provided in order to invoke all interfaces except the session interfaces.

U.2.3.3 Parameter GS_Transaction_ID

The parameter GS_Transaction_ID uniquely identifies request and response portions of a transaction within the context of a specific session.

The transaction identifier is used to match a GIAP interface request with the corresponding GIAP interface response.

U.2.3.4 Parameter GS_Lease_ID

GS_Lease_ID identifies gateway entity resources and communication resources that are allocated to a particular session.

The lease identifier is provided by the GIAP interface user when an interface is invoked to identify the particular communication resources used to support the interface.

U.2.3.5 Parameter GS_Status

GS_Status is returned by a confirm primitive. It may represent either the status resulting from handling a local request or the status corresponding to a response received from a remote entity.

The status indicates the success or failure of the interface call and, when applicable, the reason for failure.

U.2.3.6 Parameter GS_Network_Address

GS_Network_Address is an IPv6Address used to identify a logical device that is unique across all networks.

This parameter uniquely identifies a specific NLE.

U.2.3.7 Parameter GS_Unique_Device_ID

GS_Unique_Device_ID is a an EUI64Address.

This parameter uniquely identifies a specific physical device for asset management purposes.

U.2.3.8 Parameter GS_Network_ID

GS_Network_ID is a unique identifier for one of several networks that may be accessible through a single gateway.

This parameter uniquely identifies a specific network.

U.2.3.9 Parameter GS_Time

GS_Time is a 48-bit TAI time field.

The time parameter is used to describe time-related fields such as timestamp, start time, and stop time.

U.2.3.10 Parameter GS_Transfer_Mode

GS_Transfer_Mode identifies GPDU-level transfer variations.

GS_Transfer_Mode is provided with each GPDU provided for transfer in order to indicate the desired quality of service and the priority associated with the transfer of the PDUs generated to support the notional interface primitive.

U.2.4 Detailed description of interfaces

U.2.4.1 Session management interface

U.2.4.1.1 General

A gateway entity is a process within a gateway that provides gateway interfaces through the GIAP. A gateway-internal client is a user of gateway entity-provided interfaces. Typical gateway-internal clients include host systems, asset management systems, and engineering tools.

Gateway entities provide interfaces to gateway-internal clients within the context of a session. The session management interface is used to establish and manage these sessions. All other gateway entity-provided interfaces are used within the context of an established session.

A foreign protocol translator within the gateway may establish sessions on behalf of remote clients and perform protocol translation on the communication flows that correspond to gateway entity-provided interfaces.

The primary purpose of a session is to allow resource allocation and bulk reclamation of gateway and communication resources on a per-gateway entity client basis.

A session may be established by a local process or remotely (such as through a TCP/IP remote session).

One or more sessions may exist concurrently between a gateway entity and one or more gateway-internal clients. Each session is uniquely identified.

NOTE The number of concurrent sessions supported is implementation-dependent. It is possible that some implementations provide a fixed function gateway with a single session, while other implementations provide a number of sessions that are allocated on demand to a variety of applications, including host systems, historians, asset management tools, and engineering tools.

The gateway-internal client uses the G_Session primitive to create, renew or delete a session.

U.2.4.1.2 G_Session primitive

U.2.4.1.2.1 Primitives and their parameters

Table U.2 describes parameter usage for the primitive G_Session.

Table U.2 – Primitive G_Session parameter usage

Parameter name	G_Session	
	Request	Confirm
GS_Session_ID	M	M
GS_Session_Period	M	M
GS_Network_ID	M	—
GS_Status	—	M

U.2.4.1.2.2 Use of G_Session request

The gateway-internal client uses the primitive G_Session request to create, renew or delete a session.

A session is created by providing a null session identifier (GS_Session_ID = 0) and a requested session duration. Submitting GS_Session_Period > 0 requests a limited duration session, specified in seconds. Submitting GS_Session_Period = -1 requests an indefinite session duration.

A limited duration session is renewed by providing an existing (non-null) session identifier and session duration greater than 0 s. An indefinite duration session does not need to be renewed.

Changing a limited duration session to an indefinite duration session by attempting to renew it with a specified duration of -1 s is not permitted.

NOTE The upper bound of session duration is implementation-dependent. For instance, implementations are able to dedicate resources to specific applications, such as a host system, never releasing those resources.

A session is deleted by providing an existing (non-null) session identifier and session duration of 0 s.

A gateway may connect to multiple networks. Each session is associated with a specific network. A network identifier (GS_Network_ID) is specified to establish a particular network for the session. The scope of further identifiers used within a session is limited to the particular network.

U.2.4.1.2.3 Use of G_Session confirm

The gateway entity uses the G_Session confirm primitive to complete the G_Session request to the gateway-internal client.

For a successful session creation request, the gateway entity returns a unique, non-null session identifier. This identifier is used in subsequent session renew and delete operations. GS_Session_Period is returned with the actual session duration allocated by the gateway entity.

The GS_Session_Period value returned may not be the same as the value requested.

For a session renew request, the request session identifier is echoed, and a new session duration is returned.

For a session deletion request, the session identifier is echoed, and the session duration is set to 0 s.

GS_Status is returned to indicate the success or failure of the operation, as described in Table U.3.

Table U.3 – GS_Status for G_Session confirm

Value	Meaning
0	Success; new session created, renewed or deleted
1	Success; new session created or renewed with reduced period
2	Failure; session does not exist to renew or delete
3	Failure; session cannot be created (no additional sessions available) This may occur, for example, if sessions have expired, but have not explicitly been deleted
4	Failure; other

U.2.4.2 Lease management interface

U.2.4.2.1 General

Gateway entities allocate communication resources to gateway-internal clients via leases. The lease management interface is used to establish and manage leases.

The primary purpose of a lease is to allow fine-grained communication resource allocation and reclamation on a per-session basis.

Resources may be separately allocated depending on communication needs. For example, client/server, publish/subscribe, bulk transfer, and alert subscription resources may be separately allocated.

One or more leases may exist concurrently between a gateway entity and one or more gateway-internal clients. Each lease is uniquely identified within a session.

U.2.4.2.2 Use of the interface

The gateway-internal client uses the G_Lease primitive to create, renew or delete a lease.

U.2.4.2.3 G_Lease primitive

U.2.4.2.3.1 Primitives and their parameters

Table U.4 describes parameter usage for the primitive G_Lease.

Table U.4 – Primitive G_Lease parameter usage

Parameter name	G_Lease	
	Request	Confirm
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Lease_ID	M	M
GS_Lease_Period	M	M
GS_Lease_Type	M	—
GS_Protocol_Type	M	—
GS_Network_Address_List	C	—
GS_Network_Address	C	—
GS_Resource	C	—
GS_Lease_Parameters	C	—
GS_Transfer_Mode	C	—
GS_Update_Policy	C	—
GS_Period	C	—
GS_Phase	C	—
GS_Stale_Limit	C	—
GS_Connection_Info	C	—
GS_Wireless_Parameters	C	—
GS_Status	—	M

U.2.4.2.3.2 Use of G_Lease request

The gateway-internal client uses the primitive G_Lease request to create, renew, or delete a lease.

A session identifier (GS_Session_ID) is included in the G_Lease request primitives.

A session unique transaction identifier (GS_Transaction_ID) is specified for each invocation of the interface.

A lease is created by providing a null lease identifier (GS_Lease_ID = 0) and a requested lease duration. Submitting GS_Lease_Period > 0 requests a limited duration lease, specified in seconds. Submitting GS_Lease_Period = 0 requests an indefinite lease duration.

A limited duration lease is renewed by providing an existing (non-null) lease identifier and lease duration greater than 0 s. An indefinite duration lease does not need to be renewed. A limited duration lease cannot be changed to an indefinite duration lease by renewal with duration of 0 s.

The maximum supported value is implementation-dependent. Implementations can choose to dedicate resources to specific applications, such as a host system, never releasing those resources.

A lease is deleted by providing an existing lease identifier and a requested lease duration of 0 s.

Different types of leases are available, as specified by GS_Lease_Type and as shown in Table U.5. Each lease type allocates lease-specific gateway entity resources and communication resources on behalf of the gateway-internal client.

Table U.5 – GS_Lease_Type for G_Lease request

Value	Meaning
0	Client
1	Server
2	Publisher
3	Subscriber
4	Bulk transfer client
5	Bulk transfer server
6	Alert subscription

GS_Protocol_Type identifies the protocol that is associated with the lease, as indicated in Annex M. Specification of the protocol type allows special processing for particular protocols within the gateway entity.

All leases relate to establishing communication interfaces between the gateway entity and one or more device specified by one or more elements in GS_Network_Address_List. Alert subscription leases do not allocate communication resources during lease establishment, but dynamically as alert subscriptions are modified. GS_Network_Address_List is not used with the alert subscription lease type.

Client, server, subscriber, bulk transfer client, and bulk transfer server describe only a single element within GS_Network_Address_List. The publisher lease type may describe multiple elements.

The specification of multiple IPv6Addresses within the GS_Network_Address_List represents a multicast group. Elements within the G_Network_Address_List include GS_Network_Address.

GS_Lease_Parameters is a parameter structure for the specification of parameters necessary for the establishment of certain lease types. Usage of the GS_Lease_Parameters is conditioned on the specific lease type as follows.

NOTE Annex P provides additional information on detailed GS_Lease_Parameters usage.

Client, server, publish, and subscribe leases describe a unique GS_Resource. This value identifies matching client and server connection endpoints, and also identifies matching publisher and subscriber endpoints. GS_Resource also is specified by the bulk transfer client to identify the upload/download item.

Publisher leases require specification of GS_Update_Policy, GS_Period, GS_Phase, and GS_Stale_Limit to control timing and buffered behavior. GS_Transfer_Mode also is specified in order to set the default transfer quality of interface and priority.

Subscriber leases require specification of GS_Update_Policy, GS_Period, GS_Phase, and GS_Stale_Limit to control timing and buffered behavior.

A publisher and subscriber may agree to describe GS_Connection_Info in the subscriber lease for provision on each publication receipt.

Client and server leases describe GS_Transfer_Mode in order to set the default transfer quality of interface and priority.

An additional GS_Wireless_Parameters field usage depends on gateway construction. This allows access to all exposed, requestable communication features.

U.2.4.2.3.3 Use of G_Lease confirm

The gateway entity uses the primitive G_Lease confirm to complete the G_Lease request to the gateway-internal client.

The session identifier (GS_Session_ID) and transaction identifier (GS_Transaction_ID) are returned to allow matching of the confirm primitive with the original request primitive.

For a successful lease create request, the gateway entity returns a session unique lease identifier. This lease identifier is used in subsequent lease renew and delete operations. GS_Lease_Period is returned with the actual lease duration allocated by the gateway entity.

For a lease renew request, the request lease identifier is echoed, and the actual lease duration is given.

For a lease delete request, the lease identifier is echoed, and the lease duration is set to 0 s.

GS_Status is returned to indicate success or failure of the operation, as described in Table U.6.

Table U.6 – GS_Status for G_Lease confirm

Value	Meaning
0	Success; new lease created, renewed or deleted
1	Success; new lease created or renewed with reduced period
2	Failure; lease does not exist to renew or delete
3	Failure; no additional leases available
4	Failure; no device exists at IPv6Address
5	Failure; invalid lease type
6	Failure; invalid lease type information
7	Failure; other

U.2.4.3 Device list report interface

U.2.4.3.1 General

The device list report interface provides a report of the devices that are associated with a gateway. This is useful for mapping wireless devices to host systems and network browsers.

The gateway-internal client uses the G_Device_List_Report primitive to retrieve a report on the devices associated with a gateway entity.

U.2.4.3.2 G_Device_List_Report primitive

U.2.4.3.2.1 Primitives and their parameters

Table U.7 describes parameter usage for the primitive G_Device_List_Report.

Table U.7 – Primitive G_Device_List_Report parameter usage

Parameter name	G_Device_List_Report	
	Request	Confirm
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Device_List	—	M
GS_Network_Address	—	M
GS_Device_Type	—	M
GS_Unique_Device_ID	—	M
GS_Manufacturer	—	M
GS_Model	—	M
GS_Revision	—	M
GS_Status	—	M

The gateway-internal client uses the primitive G_Device_List_Report request to retrieve a report on the devices associated with a gateway entity.

A session identifier (GS_Session_ID) is obtained from the G_Session interface and included in the request.

A session unique transaction identifier (GS_Transaction_ID) is specified for each invocation of the interface.

U.2.4.3.2.2 Use of G_Device_List_Report confirm

The gateway entity uses the primitive G_Device_List_Report confirm to complete the G_Device_List_Report request to the gateway-internal client.

The session identifier (GS_Session_ID) and transaction identifier (GS_Transaction_ID) are returned to allow matching of the confirm with the original request.

A list of devices associated with the gateway entity (GS_Device_List) is returned. For each device, the list includes the IPv6Address (GS_Network_Address), the type of the device (GS_Device_Type), and the unique device identifier (GS_Unique_Device_ID).

The list also includes additional manufacturer related information (GS_Manufacturer, GS_Model, and GS_Revision).

Where the gateway includes the role of the provisioning device, the IPv6Address may be a default address. The unique identifier and the manufacturer information are used within the host-level applications to control device commissioning through the device configuration interface.

For example, a browser may display a list of devices available for provisioning along with identification information. A select set of the devices are picked from the display and are commissioned with the IPv6Address and other information to join the system. The browsing display is then refreshed with devices that are available for linkage to a control strategy.

GS_Status is returned to indicate success or failure of the operation, as described in Table U.8.

Table U.8 – GS_Status for G_Device_List_Report confirm

Value	Meaning
0	Success
1	Failure

U.2.4.4 Topology report interface

U.2.4.4.1 General

In system configurations where access to system management information is via the gateway, the topology report interface provides a topology report that relates devices within a wireless mesh.

The gateway-internal client uses the G_Topology_Report primitive to retrieve a report on the devices associated with a gateway entity.

U.2.4.4.2 G_Topology_Report primitive

U.2.4.4.2.1 Primitives and their parameters

Table U.9 describes parameter usage for the primitive G_Topology_Report.

Table U.9 – Primitive G_Topology_Report parameter usage

Parameter name	G_Topology_Report	
	Request	Confirm
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Device_List	—	M
GS_Network_Address	—	M
GS_Neighbor_List	—	M
GS_Network_Address	—	M
GS_Graph_List	—	M
GS_Graph_ID	—	M
GS_Network_Address	—	M
GS_Status	—	M

U.2.4.4.2.2 Use of G_Topology_Report request

The gateway-internal client uses the primitive G_Topology_Report request to retrieve a report on the topology of the devices associated with a gateway entity.

A session identifier (GS_Session_ID) is obtained from the G_Session interface and included in the request.

A session unique transaction identifier (GS_Transaction_ID) is specified for each invocation of the interface.

U.2.4.4.2.3 Use of G_Topology_Report confirm

The gateway entity uses the primitive G_Topology_Report confirm to complete the G_Topology_Report request to the gateway-internal client.

The session identifier (GS_Session_ID) and transaction identifier (GS_Transaction_ID) are returned to allow matching of a confirm with the original request.

A list of devices associated with the gateway entity (GS_Device_List) is returned. The list includes the IPv6Address (GS_Network_Address) for each device.

Also included within the list is a second list (GS_Neighbor_List) of the neighbor devices associated with each device. The list includes the IPv6Address (GS_Network_Address) of all neighbors (as described in the neighbor health report).

Also included within the list is a third list (GS_Graph_List) of the graph connections associated with each device. For each graph connection, the list includes the graph identified (GS_Graph_ID) and an associated IPv6Address list (GS_Network_Address) of the neighbors on the graph.

GS_Status is returned to indicate success or failure of the operation, as described in Table U.8.

U.2.4.5 Schedule report interface

U.2.4.5.1 General

In system configurations where access to system management information is via the gateway, the schedule report interface provides a schedule report detailing time slot and channel allocations on a per-device basis.

The gateway-internal client uses the G_Schedule_Report primitive to retrieve a report on the schedule of the devices associated with a gateway entity.

U.2.4.5.2 G_Schedule_Report primitive

U.2.4.5.2.1 Primitives and their parameters

Table U.10 describes parameter usage for the primitive G_Schedule_Report.

Table U.10 – Primitive G_Schedule_Report parameter usage

Parameter name	G_Schedule_Report	
	Request	Confirm
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Network_Address	M	—
GS_Channel_List	—	M
GS_Channel_Number	—	M
GS_Channel_Status	—	M
GS_Device_Schedule	—	M
GS_Network_Address	—	M
GS_Superframe_List	—	C
GS_Superframe_ID	—	C
GS_Num_Time_Slots	—	C
GS_Start_Time	—	C
GS_Link_List	—	C
GS_Network_Address	—	C
GS_Slot_Size	—	C
GS_Channel	—	C
GS_Direction	—	C
GS_Link_Type	—	C
GS_Status	—	M

U.2.4.5.2.2 Use of G_Schedule_Report request

The gateway-internal client uses the primitive G_Schedule_Report request to retrieve a schedule report for a specific device associated with a gateway entity. The particular device is identified by its IPv6Address (GS_Network_Address).

A session identifier (GS_Session_ID) is obtained from the G_Session interface and included in the request.

A session unique transaction identifier (GS_Transaction_ID) is specified for each invocation of the interface.

U.2.4.5.2.3 Use of G_Schedule_Report confirm

The gateway entity uses the primitive G_Schedule_Report confirm to complete the G_Schedule_Report request to the gateway-internal client.

The session identifier (GS_Session_ID) and transaction identifier (GS_Transaction_ID) are returned to allow matching of the confirm with the original request.

A list of channels is returned. Each element of the list includes the channel number (GS_Channel_Number) and the status of the channel (GS_Channel_Status). Channel status is set to 0 to indicate a disabled channel or to 1 to indicate an enabled channel.

A device schedule (GS_Device_Schedule) is also returned. The schedule includes device identification information (GS_Network_Address) and a list of superframes (GS_Superframe_List) that are used by the device for communication.

If a device does not use superframes, GS_Superframe_List is not returned.

The superframe list includes general superframe information, including a superframe identifier (GS_Superframe_ID), the number of time slots in the superframe (GS_Num_Time_Slots), and the start time of the superframe (GS_Start_Time). Only active superframes are reported.

GS_Start_Time is an offset relative to the beginning of TAI time. This number has significance only relative to the current network time, unless the communication is synchronized to an external source. GS_Start_Time is set to -1 to indicate that the superframe has no known synchronization.

The superframe list also includes an ordered list (GS_Link_List) with one element per timeslot in the superframe. The link list elements are used to describe communication relationships related to the superframe. Each link list element describes the timeslot duration in microseconds (GS_Slot_Size) and the channel (GS_Channel) for communication within the superframe. The link (GS_Direction) parameter describes the direction of communications. A value of 0 describes reception, and a value of 1 describes transmission. The IPv6Address (GS_Network_Address) describes the logical IPv6Address of a communication partner for the slot.

The link type (GS_Link_Type) describes the purpose of the communication:

- A value of 0 describes aperiodic data communication.
- A value of 1 describes aperiodic management communication.
- A value of 2 describes periodic data communication.
- A value of 3 describes periodic management communication.

GS_Status is returned to indicate success or failure of the operation, as described in Table U.8.

U.2.4.6 Device health report interface

U.2.4.6.1 General

The device health report interface provides a communication health report for each device's view of its own health.

The gateway-internal client uses the G_Device_Health_Report primitive to retrieve a device health report for a specified set of devices that are associated with a gateway entity.

U.2.4.6.2 G_Device_Health_Report primitive

U.2.4.6.2.1 Primitives and their parameters

Table U.11 describes parameter usage for the primitive G_Device_Health_Report.

Table U.11 – Primitive G_Device_Health_Report parameter usage

Parameter name	G_Device_Health_Report	
	Request	Confirm
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Device_List	M	M
GS_Network_Address	M	M(=)
GS_DPDUs_Transmitted	—	M
GS_DPDUs_Received	—	M
GS_DPDUs_Failed_Transmission	—	M
GS_DPDUs_Failed_Reception	—	M
GS_Status	—	M

U.2.4.6.2.2 Use of G_Device_Health_Report request

The gateway-internal client uses the primitive G_Device_Health_Report request to retrieve a device health report for a specified set of devices that are associated with a gateway entity.

A session identifier (GS_Session_ID) is obtained from the G_Session interface and included in the request.

A session unique transaction identifier (GS_Transaction_ID) is specified for each invocation of the interface.

A health report is requested for a specific list of devices (GS_Device_List). The IPv6Address of each device (GS_Network_Address) is required.

U.2.4.6.2.3 Use of G_Device_Health_Report confirm

The gateway entity uses the primitive G_Device_Health_Report confirm to complete the G_Device_Health_Report request to the gateway-internal client.

The session identifier (GS_Session_ID) and transaction identifier (GS_Transaction_ID) are returned to allow matching of the confirm with the original request.

A device list (GS_Device_List) is returned. The list includes device identification information (GS_Network_Address) and communication health information. The communication health information includes the total number of DPDUs transmitted (GS_DPDUs_Transmitted) from the device to all neighbors, the total number of DPDUs received (GS_DPDUs_Received) from the device by all neighbors, the total number of DPDUs to all neighbors that failed transmission (GS_DPDUs_Failed_Transmission), and the total number of DPDUs from all neighbors that failed reception (GS_DPDUs_Failed_Reception). Failed receptions include identifiable DPDUs that are discarded due to transmission-related corruption.

NOTE Failed receptions will likely be less than failed transmissions, since many failed DPDUs will not have enough uncorrupted information to determine the addressing. Failed reception does not include protocol-related errors.

GS_Status is returned to indicate success or failure of the operation, as described in Table U.8.

U.2.4.7 Neighbor health report interface

U.2.4.7.1 General

In system configurations where access to system management information is via the gateway, the neighbor health report interface provides a communication health report for each device's view of its neighbors.

A neighbor device is a link level wireless communication partner that is configured for direct exchange of DPDUs (RF transmission without hops). The neighbor health report interfaces provide information on these physical neighbors. Neighbor devices are able to collect DPDU exchange statistics that indicate local RF conditions.

The gateway-internal client uses the G_Neighbor_Health_Report primitive to retrieve a communication health report for the set of neighbor devices associated with a specific device that is associated with a gateway entity.

U.2.4.7.2 G_Neighbor_Health_Report primitive

U.2.4.7.2.1 Primitives and their parameters

Table U.12 describes parameter usage for the primitive G_Neighbor_Health_Report.

Table U.12 – Primitive G_Neighbor_Health_Report parameter usage

Parameter name	G_Neighbor_Health_Report	
	Request	Confirm
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Network_Address	M	—
GS_Neighbor_Health_List	—	M
GS_Network_Address	—	M
GS_Link_Status	—	M
GS_DPDUs_Transmitted	—	M
GS_DPDUs_Received	—	M
GS_DPDUs_Failed_Transmission	—	M
GS_DPDUs_Failed_Reception	—	M
GS_Signal_Strength	—	M
GS_Signal_Quality	—	M
GS_Status	—	M

U.2.4.7.2.2 Use of G_Neighbor_Health_Report request

The gateway-internal client uses the primitive G_Neighbor_Health_Report request to retrieve a communication health report for the set of neighbor devices associated with a specific device that is associated with a gateway entity.

A session identifier (GS_Session_ID) is obtained from the G_Session interface and included in the request.

A session unique transaction identifier (GS_Transaction_ID) is specified for each invocation of the interface.

A neighbor health report is requested for a device at a specific IPv6Address (GS_Network_Address).

U.2.4.7.2.3 Use of G_Neighbor_Health_Report confirm

The gateway entity uses the primitive G_Neighbor_Health_Report confirm to complete the G_Neighbor_Health_Report request to the gateway-internal client.

The session identifier (GS_Session_ID) and transaction identifier (GS_Transaction_ID) are returned to allow matching of the confirm with the original request.

A neighbor health list (GS_Neighbor_Health_List) is returned. The list includes the neighbor device identification information (GS_Network_Address) and communication health information. The communication health information includes a general status (GS_Link_Status). GS_Link_Status = 1 indicates that the neighbor is available for communication. GS_Link_Status = 0 indicates that the neighbor is unavailable for communication.

Health information also includes the number of DPDUs transmitted to the neighbor (GS_DPDUs_Transmitted), the number of DPDUs received from the neighbor (GS_DPDUs_Received), the number of failed transmission attempts (GS_DPDUs_Failed_Transmission), and the number of failed receptions (GS_DPDUs_Failed_Reception) from the neighbor. Failed receptions include identifiable DPDUs that are discarded due to transmission related corruption.

NOTE Failed receptions will likely be less than failed transmissions, since many failed DPDUs will not have enough uncorrupted information to determine the addressing. Failed reception does not include protocol-related errors.

Health information also includes GS_Signal_Strength and GS_Signal_Quality. These parameters return values between 0 (worst signal) and 100 (best signal). GS_Signal_Strength indicates the average uncorrelated power level of the signals received from a specific neighbor relative to the range of the receiver. GS_Signal_Quality indicates the average correlated power level of the signals received from a specific neighbor relative to the range of the receiver.

GS_Status is returned to indicate success or failure of the operation, as described in Table U.8.

U.2.4.8 Network health report interface

U.2.4.8.1 General

In system configurations where access to system management information is via the gateway, the neighbor health report interface provides a communication health report for each device's view of its neighbors.

The gateway-internal client uses the G_Network_Health_Report primitive to retrieve a summary communication health report for an entire network.

U.2.4.8.2 G_Network_Health_Report primitive

U.2.4.8.2.1 Primitives and their parameters

Table U.13 describes parameter usage for the primitive G_Network_Health_Report.

Table U.13 – Primitive G_Network_Health_Report parameter usage

Parameter name	G_Network_Health_Report	
	Request	Confirm
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Network_Health	—	M
GS_Network_ID	—	M
GS_Network_Type	—	M
GS_Device_Count	—	M
GS_Start_Date	—	M
GS_Current_Date	—	M
GS_DPDUs_Sent	—	M
GS_DPDUs_Lost	—	M
GS_GPDU_Latency	—	M
GS_GPDU_Path_Reliability	—	M
GS_GPDU_Data_Reliability	—	M
GS_Join_Count	—	M
GS_Device_Health_List	—	M
GS_Network_Address	—	M
GS_Start_Date	—	M
GS_Current_Date	—	M
GS_DPDUs_Sent	—	M
GS_DPDUs_Lost	—	M
GS_GPDU_Latency	—	M
GS_GPDU_Path_Reliability	—	M
GS_GPDU_Data_Reliability	—	M
GS_Join_Count	—	M
GS_Status	—	M

U.2.4.8.2.2 Use of G_Network_Health_Report request

The gateway-internal client uses the primitive G_Network_Health_Report request to retrieve a summary communication health report for an entire network.

A session identifier (GS_Session_ID) is obtained from the G_Session interface and included in the request.

A session unique transaction identifier (GS_Transaction_ID) is specified for each invocation of the interface.

U.2.4.8.2.3 Use of G_Network_Health_Report confirm

The gateway entity uses the primitive G_Network_Health_Report confirm to complete the G_Network_Health_Report request to the gateway-internal client.

The session identifier (GS_Session_ID) and transaction identifier (GS_Transaction_ID) are returned to allow matching of the confirm with the original request.

A network health summary (GS_Network_Health) is returned. The summary includes network identification information (GS_Network_ID and GS_Network_Type) and network communication health summary information. The communication health information includes the number of devices in the network (GS_Device_Count), the start date and the current date for the network (GS_Start_Date and GS_Current_Date), transmission statistics (GS_DPDUs_Sent, GS_DPDUs_Lost, and GS_GPDU_Latency), reliability statistics (GS_GPDU_Path_Reliability and GS_GPDU_Data_Reliability), and join statistics (GS_Join_Count).

A device-specific health summary (GS_Device_Health_List) is also returned. The list includes device identification information (GS_Network_Address) and communication statistics that are an identical subset of those contained in the network health summary (GS_Start_Date, GS_Current_Date, GS_DPDUs_Sent, GS_DPDUs_Lost, GS_GPDU_Latency, GS_GPDU_Path_Reliability, GS_GPDU_Data_Reliability, and GS_Join_Count).

GS_Start_Date is a 48-bit TAI time field indicating the time when a device first started operating. This is useful for calculation of battery replacement schedules.

GS_Current_Date is a 48-bit TAI time field indicating the current time as viewed by the device. This is the time used by the device for timestamp purposes.

GS_GPDU_Latency is a number from 0..100 indicating the percentage of scheduled GPDUs that arrive later than expected. These GPDUs may be delayed due to delivery over secondary paths or due to congestion in intermediate devices.

GS_GPDU_Path_Reliability is a number from 0..100 indicating the percentage of first path success for acknowledged GPDU transmission. GPDUs that are transmitted on a secondary path may arrive successfully, but may reduce the path reliability.

GS_GPDU_Data_Reliability is a number from 0..100 indicating the percentage of total GPDUs that are successful GPDUs. The total GPDUs are the number of acknowledged transmit GPDUs that are attempted plus the number of received GPDUs. Successful GPDUs are acknowledged transmit GPDUs that are transferred correctly on the first attempt plus receive GPDUs that pass integrity checks.

GS_Join_Count is a positive integer that indicates the number of times a device has joined the system. Join count may rise if power is interrupted, a device is reset, the network is reformed, or a device is moved to a new network. Excessive joins may indicate device integrity or communication problems.

GS_Status is returned to indicate success or failure of the operation, as described in Table U.8.

U.2.4.9 Time interface

U.2.4.9.1 General

The time interface enables retrieval and setting of the time for a wireless network associated with a gateway. This is useful for time synchronization of a network of wireless devices with a host system and other host-level applications.

The gateway-internal client uses the G_Time primitive to retrieve a report on the devices associated with a gateway entity.

U.2.4.9.2 G_Time primitive

U.2.4.9.2.1 Primitives and their parameters

Table U.14 describes parameter usage for the primitive G_Time.

Table U.14 – Primitive G_Time parameter usage

Parameter name	G_Time	
	Request	Confirm
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Command	M	—
GS_Time	C	M
GS_Status	—	M

U.2.4.9.2.2 Use of G_Time request

The gateway-internal client uses the primitive G_Time request to read or set the network time.

A session identifier (GS_Session_ID) is obtained from the G_Session interface and included in the request.

A session unique transaction identifier (GS_Transaction_ID) is specified for each invocation of the interface.

GS_Command = 0 reads the network time. GS_Time is not included.

GS_Command = 1 attempts to set the network time. A new time (GS_Time) is provided.

U.2.4.9.2.3 Use of G_Time confirm

The gateway entity uses the primitive G_Time confirm to complete the G_Time request to the gateway-internal client.

The session identifier (GS_Session_ID) and transaction identifier (GS_Transaction_ID) are returned to allow matching of the confirm with the original request.

If GS_Command = 0 in the request, the current network time (GS_Time) is returned.

If GS_Command = 1 in the request, the interface attempts to set the network time. The current network time (GS_Time) is returned. If the update is successful, the current time will reflect the change.

GS_Status is returned to indicate success or failure of the operation, as described in Table U.15.

Table U.15 – GS_Status for G_Time confirm

Value	Meaning
0	Success
1	Failure; not allowed to set time in this configuration
2	Failure; other

U.2.4.10 Client/server interface

U.2.4.10.1 General

The client/server interface provides for client/server data transfer. The necessary communication resources to enable the transfer are allocated through the use of the lease interface. The client and server each perform separate but related roles. Linkage of the client and the server is accomplished through the establishment of leases with matching lease information. Communication resources include local buffer facilities in order to minimize energy consuming transactions. Clients and servers may exist either in the gateway or in devices.

The G_Client_Server primitive is used to send an internal client request data payload to a server and to initiate receipt of a corresponding server response data payload. Depending on implementation, the response payload may come from the internal client buffer within the gateway or from the field device.

U.2.4.10.2 G_Client_Server primitive

U.2.4.10.2.1 Primitives and their parameters

Table U.16 describes parameter usage for the primitive G_Client_Server.

Table U.16 – Primitive G_Client_Server parameter usage

Parameter name	G_Client_Server			
	Request	Indication	Response	Confirm
GS_Session_ID	M	—	—	M(=)
GS_Transaction_ID	M	—	—	M(=)
GS_Lease_ID	M	—	—	—
GS_Buffer	M	—	—	—
GS_Transfer_Mode	M	—	M	—
GS_Request_Data	M	C(=)	—	—
GS_Response_Data	—	—	C	M
GS_Transaction_Info	C	—	—	C(=)
GS_Status	—	—	M	M

U.2.4.10.2.2 Use of G_Client_Server request

The primitive G_Client_Server request is used to either attempt to acquire the requested data locally from the gateway (if GS_Buffer = 1), or to send a corresponding WISN native client application data request using the content of the data payload (GS_Request_Data) parameter to a WISN communicating device and to initiate receipt of a corresponding WISN server response. Whether the requested data is accessed locally or remotely, the data is returned via the response/confirm primitive parameter for the response payload (GS_Response_Data).

A session identifier (GS_Session_ID) is obtained from the G_Session interface and included in the request.

A session unique transaction identifier (GS_Transaction_ID) is specified for each invocation of the interface.

The server device is known through the lease identifier (GS_Lease_ID) that was obtained from the lease interface.

The response data will be requested from the server device if the buffer is disabled for the transaction (GS_Buffer = 0). The response data will be delivered from the buffer if the buffer is enabled for the transaction (GS_Buffer = 1) and the buffer contains a matching response that has not expired.

GS_Transfer_Mode is provided with the request in order to indicate the quality of interface and priority for the transfer of the data.

If GS_Transaction_Info is provided as part of a request, it is returned by the corresponding confirm primitive.

U.2.4.10.2.3 Use of G_Client_Server indication

The primitive G_Client_Server indication is used to signal the arrival of a client request data payload at the server for processing.

The indication is conditional on whether the server response data payload could be delivered from the client buffer.

U.2.4.10.2.4 Use of G_Client_Server response

The primitive G_Client_Server response is used to return a server response data payload to the client.

GS_Transfer_Mode is provided with the response in order to indicate the quality of interface and priority for the transfer of the data.

The response is conditional on whether the server response data payload could be delivered from the client buffer.

U.2.4.10.2.5 Use of G_Client_Server confirm

The primitive G_Client_Server confirm is used to complete the G_Client_Server request to the client.

The session identifier (GS_Session_ID) and transaction identifier (GS_Transaction_ID) are returned to allow matching of the confirm primitive with the original request primitive.

A server response data payload is returned. The payload is either delivered from the client buffer or from the server.

If GS_Transaction_Info was provided in the request, it will be returned in the confirm.

GS_Status is returned to indicate success or failure of the operation, as described in Table U.17.

Table U.17 – GS_Status for G_Client_Server confirm

Value	Meaning
0	Success
1	Failure; server is inaccessible for unbuffered request
2	Failure; server is inaccessible and client buffer is invalid for buffered request
3	Failure; lease has expired
4	Failure; other

U.2.4.11 Publish/subscribe interface

U.2.4.11.1 General

The publish/subscribe interface provides mechanisms for publish/subscribe data transfer. The necessary communication resources to enable message exchange are allocated through the use of the lease interface. The publisher and the subscriber each perform separate but related roles. Linkage of the publisher and the subscriber is accomplished through separate establishment of matching communication. Communication resources include local buffer facilities in both the publisher and the subscriber in order to minimize energy consuming transactions. Publishers and subscribers may exist in gateways, adapters, or native devices.

U.2.4.11.2 Lease establishment

The G_Lease interface is used prior to the use of the G_Publish interface in order to establish a GS_Lease_ID. The GS_Lease_Type is set to either publisher or subscriber to configure the respective side and to establish and reserve the underlying gateway entity and communication channel resources.

GS_Network_Address_List is used by the publisher and subscriber to establish the identity of the other endpoints. A publisher may describe multiple addresses within the list in order to configure multiple subscribers.

Within the lease, GS_Protocol_Type is used to describe the application protocol that will be tunneled through the interface. This allows protocol-specific processing to occur.

GS_Lease_Parameters is used to establish the expected protocol interaction between a publisher and a subscriber.

U.2.4.11.3 Publication

The G_Publish primitive is used by a publisher to initiate transfer of a publish data payload to one or more subscribers.

The publish data payload is stored in a local buffer and forwarded from the buffer to subscribers. The lease configuration parameters determine when forwarding will occur. Forwarding occurs in order to meet scheduled deadlines. Over a period of time, the same payload may be forwarded multiple times to indicate that the publisher still exists and to prevent timeout. Invocation with unchanged data may not result in forwarding.

U.2.4.11.4 Subscription

The G_Subscribe primitive is used by a subscriber to retrieve the most recent publication data from the local buffer.

The subscriber also receives the most recent publication associated with a subscribe lease via the G_Publish indication primitive.

The primitive G_Publish_Watchdog is used within a subscriber to signal the expiration of a watchdog timer. The timer expires in the absence of expected updates from a publisher. The timer is reset on the arrival of publication data payload at the subscriber. The watchdog timer is configured as part of the lease configuration parameters.

The primitive G_Publish_Timer is used within a publisher to signal the expiration of a publication timer. The publication timer is a periodic timer that expires prior to the deadline for forwarding the publish data payload. The indication may be used to publish fresh data. The publication timer is configured with the lease configuration parameters.

The primitive G_Subscribe_Timer is used within a subscriber to signal the expiration of a subscription timer. The subscription timer is a periodic timer that expires at the delivery deadline for receiving the publish data payload. The indication is used to process existing publication data. Arrival of fresh data will reset the timer and will result in a G_Publish indication. The subscription timer is configured with the lease configuration parameters.

U.2.4.11.5 Types of primitives and parameters

U.2.4.11.5.1 G_Publish primitive and its parameters

Table U.18 describes parameter usage for the primitive G_Publish.

Table U.18 – Primitive G_Publish parameter usage

Parameter name	G_Publish		
	Request	Indication	Confirm
GS_Session_ID	M	M	M
GS_Transaction_ID	M	—	M(=)
GS_Lease_ID	M	M	—
GS_Transfer_Mode	M	—	—
GS_Publish_Data	M	C(=)	—
GS_Status	—	—	M

U.2.4.11.5.2 Use of G_Publish request

The primitive G_Publish request is used to initiate transfer of a publish data payload (GS_Publish_Data) to one or more subscribers. The publish data payload is stored in a local buffer and forwarded from the buffer to subscribers.

A session identifier (GS_Session_ID) is obtained from the G_Session interface and included in the request.

A session unique transaction identifier (GS_Transaction_ID) is specified for each invocation of the interface.

The subscriber addressing is known through the lease identifier (GS_Lease_ID) that was obtained from the lease interface.

Within the lease parameters, GS_Resource is an identical value specified for a publisher and one or more subscribers in order to facilitate establishment of linkage between the endpoints.

GS_Transfer_Mode is provided with the request in order to indicate the quality of interface and priority for the transfer of the data.

U.2.4.11.5.3 Use of G_Publish indication

The primitive G_Publish indication is used to signal the arrival of a publish data payload at a subscriber for processing.

The publish data payload (GS_Publish_Data) is delivered with the indication.

The subscriber session identifier (GS_Session_ID) and subscriber lease identifier (GS_Lease_ID) are returned to allow association of the indication primitive with a specific publish/subscribe relationship.

The indication is conditional on configuration-dependent timed delivery from the publisher and indicates fresh publication data.

U.2.4.11.5.4 Use of G_Publish confirm

The primitive G_Publish confirm is used to complete the G_Publish request.

The publisher session identifier (GS_Session_ID) and transaction identifier (GS_Transaction_ID) are returned to allow matching of the confirm primitive with the original request primitive.

GS_Status is returned to indicate success or failure of the operation, as described in Table U.19.

Table U.19 – GS_Status for G_Publish confirm

Value	Meaning
0	Success
1	Failure; lease has expired
2	Failure; other

U.2.4.11.5.5 G_Subscribe primitive and its parameters

Table U.20 describes parameter usage for the primitive G_Subscribe.

Table U.20 – Primitive G_Subscribe parameter usage

Parameter name	G_Subscribe	
	Request	Confirm
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Lease_ID	M	—
GS_Publish_Data	—	M
GS_Status	—	M

U.2.4.11.5.6 Use of G_Subscribe request

The primitive G_Subscribe request is used to retrieve the most recent publication data (GS_Publish_Data) from the local buffer.

A session identifier (GS_Session_ID) is obtained from the G_Session interface and included in the request.

A session unique transaction identifier (GS_Transaction_ID) is specified for each invocation of the interface.

The publisher addressing is known through the lease identifier (GS_Lease_ID) that was obtained from the lease interface.

U.2.4.11.5.7 Use of G_Subscribe confirm

The primitive G_Subscribe confirm is used to complete the G_Subscribe request.

The session identifier (GS_Session_ID) and transaction identifier (GS_Transaction_ID) are returned to allow matching of the confirm primitive with the original request primitive.

GS_Status is returned to indicate success or failure of the operation, as described in Table U.21.

Table U.21 – GS_Status for G_Subscribe confirm

Value	Meaning
0	Success; fresh data
1	Success; stale data
2	Failure; lease has expired
3	Failure; other

U.2.4.11.5.8 G_Publish_Timer primitive and its parameters

Table U.22 describes parameter usage for the primitive G_Publish_Timer.

Table U.22 – Primitive G_Publish_Timer parameter usage

Parameter name	G_PublishTimer
	Indication
GS_Session_ID	M
GS_Lease_ID	M

U.2.4.11.5.9 Use of G_Publish_Timer indication

The primitive G_Publish_Timer indication is used within a publisher to signal the expiration of a publication timer.

The publisher session identifier (GS_Session_ID) and publisher lease identifier (GS_Lease_ID) are returned to allow association of the indication primitive with a specific publish/subscribe relationship.

U.2.4.11.5.10 G_Subscribe_Timer primitive and its parameters

Table U.23 describes parameter usage for the primitive G_Subscribe_Timer.

Table U.23 – Primitive G_Subscribe_Timer parameter usage

Parameter name	G_SubscribeTimer
	Indication
GS_Session_ID	M
GS_Publish_Data	M
GS_Lease_ID	M

U.2.4.11.5.11 Use of G_Subscribe_Timer indication

The primitive G_Subscribe_Timer indication is used within a subscriber to signal the expiration of a subscription timer. The timer is reset by the G_Publish indication.

The publish data payload (GS_Publish_Data) is delivered from the subscriber buffer with the indication.

The subscriber session identifier (GS_Session_ID) and subscriber lease identifier (GS_Lease_ID) are returned to allow association of the indication primitive with a specific publish/subscribe relationship.

U.2.4.11.5.12 G_Publish_Watchdog primitive and its parameters

Table U.24 describes parameter usage for the primitive G_Publish_Watchdog.

Table U.24 – Primitive G_Publish_Watchdog parameter usage

Parameter name	G_Publish_Watchdog
	Indication
GS_Session_ID	M
GS_Publish_Data	M
GS_Lease_ID	M

U.2.4.11.5.13 Use of G_Publish_Watchdog indication

The primitive G_Publish_Watchdog indication is used within a subscriber to signal the expiration of a watchdog timer due to the absence of expected updates from a publisher. The timer is reset by the G_Publish indication.

The now-stale publish data payload (GS_Publish_Data) is delivered from the subscriber buffer with the indication.

The session identifier (GS_Session_ID) and lease identifier (GS_Lease_ID) are returned to allow association of the indication primitive with a specific publish/subscribe relationship.

U.2.4.12 Bulk transfer interface

U.2.4.12.1 General

The bulk transfer interface provides for bulk data transfer. Bulk data transfer is used to transfer large items between gateway-internal clients and wireless devices.

Bulk transfers operate in the context of a session between the GIAP interface provider and the GIAP interface user. All primitives supported by the gateway through a GIAP include the corresponding GS_Session_ID.

The client of the session manages the session-unique GS_Transaction_IDs for each primitive invoked by the client. This is necessary in order to maintain coordination between bulk transfer primitives.

The GS_Lease_ID, which represents the necessary communication resources allocated within the gateway, is supplied with each primitive.

Separate parallel bulk transfers are distinguished by a GS_Transfer_ID. A GS_Transfer_ID also is included in each GIAP interface primitive. The transfer state is maintained for each bulk transfer in progress. For example, the block number being transferred is maintained by the endpoints.

G_Bulk_Open is used to open a bulk transfer. G_Bulk_Close is used to close a bulk transfer. G_Bulk_Transfer is used to perform the actual transfer of data segments within a bulk transfer.

U.2.4.12.2 Types of primitives and parameters

U.2.4.12.2.1 G_Bulk_Open primitive and its parameters

Table U.25 describes parameter usage for the primitive G_Bulk_Open.

Table U.25 – Primitive G_Bulk_Open parameter usage

Parameter name	G_Bulk_Open	
	Request	Confirm
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Lease_ID	M	—
GS_Transfer_ID	M	—
GS_Resource	M	—
GS_Mode	M	—
GS_Block_Size	M	M
GS_Item_Size	C	C
GS_Status	—	M

U.2.4.12.2.2 Use of G_Bulk_Open request

The G_Bulk_Open request primitive is used to initiate a bulk transfer. The target device for a bulk transfer is implied by the GS_Lease_ID.

The target item for a bulk transfer is identified by GS_Resource.

A transfer is directional (upload or download) and GS_Mode describes the direction of the transfer. GS_Mode = 0 describes download and GS_Mode = 1 describes upload.

The GIAP interface user sets GS_Block_Size to request a block size for the subsequent transfer phase.

The GIAP interface user sets the GS_Item_Size to request download of an item of a particular size. The item may exceed the available download limits, resulting in an error response. GS_Item_Size = 0 requests the download of an item of indeterminate size.

U.2.4.12.2.3 Use of G_Bulk_Open confirm

The G_Bulk_Open confirm primitive is used in response to the G_Bulk_Open request.

The GS_Item_Size is set by the GIAP interface provider to indicate the item size. For a download, this is the maximum item size that will be accepted. For an upload, this is the actual item size. GS_Item_Size = 0 indicates that there is no limit imposed on the item size.

The GIAP interface provider determines and returns the GS_Block_Size that will be used for the subsequent transfer phase. The block size may be reduced in size (based on available resources) from the original size requested in the GS_Bulk_Open request.

GS_Status indicates success or failure of the G_Bulk_Open, as shown in Table U.26.

Table U.26 – GS_Status for G_Bulk_Open confirm

Value	Meaning
0	Success
1	Failure; item exceeds limits
2	Failure; unknown resource
3	Failure; invalid mode
4	Failure; other

U.2.4.12.3 G_Bulk_Transfer primitive and its parameters

Table U.27 describes parameter usage for the primitive G_Bulk_Transfer.

Table U.27 – Primitive G_Bulk_Transfer parameter usage

Parameter name	G_Bulk_Transfer	
	Request	Confirm
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Lease_ID	M	—
GS_Transfer_ID	M	—
GS_Bulk_Data	C	C
GS_Status	—	M

U.2.4.12.3.1 Use of G_Bulk_Transfer request

The G_Bulk_Transfer request primitive is used to move bulk data. GS_Bulk_Data is a transfer segment that is conditionally sent to the target in the case of a download.

G_Bulk_Transfer is used as many times as required to complete the transfer of a large item. G_Bulk_Close is used by the GIAP interface user to indicate the completion of the transfer.

U.2.4.12.3.2 Use of G_Bulk_Transfer confirm

The G_Bulk_Transfer confirm primitive is used in response to the G_Bulk_Transfer request. GS_Bulk_Data is a transfer segment that is conditionally received from the target in the case of an upload.

GS_Status indicates success or failure of the G_Bulk_Transfer, as indicated in Table U.28.

Table U.28 – GS_Status for G_Bulk_Transfer confirm

Value	Meaning
0	Success
1	Failure; communication failed
2	Failure; transfer aborted
3	Failure; other

U.2.4.12.4 G_Bulk_Close primitive and its parameters

Table U.29 describes parameter usage for the primitive G_Bulk_Close.

Table U.29 – Primitive G_Bulk_Close parameter usage

Parameter name	G_Bulk_Close	
	Request	Confirm
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Lease_ID	M	—
GS_Transfer_ID	M	—
GS_Status	—	M

U.2.4.12.4.1 Use of G_Bulk_Close request

The G_Bulk_Close request primitive is used to complete a bulk transfer, and to clean up any resources or state handling necessary in the GIAP interface provider.

U.2.4.12.4.2 Use of G_Bulk_Close confirm

The G_Bulk_Close confirm primitive is used in response to the G_Bulk_Close request.

U.2.4.13 Alert interface

U.2.4.13.1 General

The alert interface provides for the establishment of alert notification events for gateway-internal clients. Additional operations may be required to collect additional information related to the alert or to respond to the alert.

Alert interfaces operate in the context of a session between the GIAP interface provider and the GIAP interface user. All primitives supported by the gateway through a GIAP include the corresponding GS_Session_ID.

The client of the session manages the session-unique outstanding GS_Transaction_IDs for each primitive it invokes. This is necessary in order to maintain coordination between alert primitives.

The GS_Lease_ID, which represents the necessary gateway entity and communication resources, is supplied with each primitive.

G_Alert_Subscription is used to subscribe to alerts either by category or by specific alerts.

U.2.4.13.2 Types of primitives and parameters

U.2.4.13.2.1 G_Alert_Subscription primitive and its parameters

Table U.30 describes parameter usage for the primitive G_Alert_Subscription.

Table U.30 – Primitive G_Alert_Subscription parameter usage

Parameter name	G_Alert_Subscription	
	Request	Confirm
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Lease_ID	M	—
GS_Subscription_List	M	C
GS_Category	C	C
GS_Network_Address	C	C
GS_Alert_Source_ID	C	C
GS_Subscribe	M	C
GS_Enable	M	C
GS_Status	—	M

U.2.4.13.2.2 Use of G_Alert_Subscription request

The G_Alert_Subscription request primitive is used to manage an alert subscription list.

GS_Subscription_List contains one or more alert subscription modification requests. Each list element can be used to modify the subscription for a particular category of alerts or to modify the subscription for a specific alert from a specific source.

List elements may describe the GS_Category to indicate subscription modification for a particular category of alerts. Alert categories include:

0 = device;

1 = network;

2 = security; and

3 = process.

GS_Network_Address and GS_Alert_Source_ID are not supplied if GS_Category is supplied.

Alternatively, list elements may describe GS_Network_Address and GS_Alert_Source_ID instead of GS_Category to describe a specific device and an identifier of a specific alert from that device.

NOTE It is anticipated that some gateway-internal clients, such as full alarm management systems, will use complete categories, whereas other gateway-internal clients are able to restrict their usage to only a select subset of alerts.

GS_Subscribe and GS_Enable control the actions for each list element. GS_Subscribe is used to describe which alerts are to be received within the gateway entity and forwarded to the GIAP interface user in the form of G_Alert_Notification indications. GS_Enable is used to control the underlying generation of alerts at the source. GS_Subscribe = 1 subscribes to a specific alert or an alert category, while GS_Subscribe = 0 unsubscribes from the alert. GS_Enable = 1 enables a specific alert or an alert category, while GS_Enable = 0 disables the alert at the source.

In order to synchronize the alarm state between the alarm source and the gateway entity, alarm recovery is initiated on subscriptions.

U.2.4.13.2.3 Use of G_Alert_Subscription confirm

The G_Alert_Subscription confirm primitive is used in response to the G_Alert_Subscription request.

GS_Status indicates success or failure of the G_Alert_Subscription request, as indicated in Table U.31. A GS_Subscription_List with a single element is conditionally returned if the operation fails. The status code relates to the particular element. Processing of the list stops at the first failed element.

Table U.31 – GS_Status for G_Alert_Subscription confirm

Value	Meaning
0	Success
1	Failure; invalid category
2	Failure; invalid individual alert
3	Failure; other

U.2.4.13.2.4 G_Alert_Notification primitive and its parameters

Table U.32 describes parameter usage for the primitive G_Alert_Notification.

Table U.32 – Primitive G_Alert_Notification parameter usage

Parameter name	G_Alert_Notification
	Indication
GS_Session_ID	M
GS_Lease_ID	M
GS_Alert	M
GS_Network_Address	M
GS_Alert_Source_ID	M
GS_Time	M
GS_Class	M
GS_Direction	M
GS_Category	M
GS_Type	M
GS_Priority	M
GS_Alert_Data	C

U.2.4.13.2.5 Use of G_Alert_Notification indication

The G_Alert_Notification indication is generated by the GIAP interface provider and sent to the GIAP user in response to an alert received by the gateway. Notification is provided only for those alerts to which the GIAP client had subscribed, and for which notification has been enabled.

A GS_Alert structure is provided within the indication to provide alert specific details as follows:

- GS_Network_Address indicates the source device of the alert.
- GS_Alert_Source_ID indicates the specific alert within the source device.

- GS_Time is a timestamp that indicates when the alert was originally generated.
- GS_Class = 0 identifies the alert as an event; GS_Class = 1 identifies the alert as an alarm.
- GS_Direction further classifies alarms as follows:

  0: alarm condition ended;

  1: alarm condition began.
- GS_Category describes the alert category as follows:

  0: process-related;

  1: device-related;

  2: network-related;

  3: security-related.
- GS_Type describes sub-categories for alerts. The actual value is application-specific.
- GS_Priority describes a priority for the alert. Larger values indicate higher priority. The actual value is application-specific.
- GS_Alert_Data allows inclusion of alert-related information. This field is conditional on whether additional alert information is available. The actual value is application-specific.

The gateway entity acknowledges the alert receipt.

U.2.4.14 Gateway configuration interface

U.2.4.14.1 General

The gateway configuration interface provides for reading and writing the gateway configuration attributes.

The gateway-internal client uses the G_Read_Gateway_Configuration primitive to retrieve gateway configuration attributes.

U.2.4.14.2 Types of primitives and parameters

U.2.4.14.2.1 G_Read_Gateway_Configuration primitive and its parameters

Table U.33 describes parameter usage for the primitive G_Read_Gateway_Configuration.

Table U.33 – Primitive G_Read_Gateway_Configuration parameter usage

Parameter name	G_ReadGatewayConfiguration	
	Request	Confirm
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Attribute_Identifier	M	—
GS_Attribute_Value	—	C
GS_Status	—	M

U.2.4.14.2.2 Use of G_Read_Gateway_Configuration request

The gateway-internal client uses the primitive G_Read_Gateway_Configuration request to retrieve gateway configuration parameters.

A session identifier (GS_Session_ID) is obtained from the G_Session interface and included in the request.

A session unique transaction identifier (GS_Transaction_ID) is specified for each invocation of the interface.

The requested attribute is specified by the attribute identifier (GS_Attribute_Identifier), as shown in Table U.34. The requested value is specified by the attribute value (GS_Attribute_Value).

Table U.34 – GS_Attribute_Identifier values for G_Read_Gateway_Configuration request

Value	Meaning
0	GS_GUID
1	GS_Max_Retries
2	GS_Max_Devices
3	GS_Actual_Devices

U.2.4.14.2.3 Use of G_Read_Gateway_Configuration confirm

The gateway entity uses the primitive G_Read_Gateway_Configuration confirm to complete the G_Read_Gateway_Configuration request to the gateway-internal client.

The session identifier (GS_Session_ID) and transaction identifier (GS_Transaction_ID) are returned to allow matching of the confirm with the original request.

If the operation succeeds, the value (GS_Attribute_Value) is returned for the requested attribute (GS_Attribute_Identifier).

GS_Status is returned to indicate success or failure of the operation, as described in Table U.8.

U.2.4.14.2.4 G_Write_Gateway_Configuration primitive and its parameters

Table U.35 describes parameter usage for the primitive G_Write_Gateway_Configuration.

Table U.35 – Primitive G_Write_Gateway_Configuration parameter usage

Parameter name	G_Write_Gateway_Configuration	
	Request	Confirm
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Attribute_Identifier	M	—
GS_Attribute_Value	M	—
GS_Status	—	M

U.2.4.14.2.5 Use of G_Write_Gateway_Configuration request

The gateway-internal client uses the primitive G_Write_Gateway_Configuration request to alter gateway configuration attributes.

A session identifier (GS_Session_ID) is obtained from the G_Session interface and included in the request.

A session unique transaction identifier (GS_Transaction_ID) is specified for each invocation of the interface.

The requested attribute is specified by the attribute identifier (GS_Attribute_Identifier), as shown in Table U.36. The requested value is specified by the attribute value (GS_Attribute_Value).

Table U.36 – GS_Attribute_Identifier values for G_Write_Gateway_Configuration request

Value	Meaning
0	GS_GUID
1	GS_Max_Retries

U.2.4.14.2.6 Use of G_Write_Gateway_Configuration confirm

The gateway entity uses the primitive G_Write_Gateway_Configuration confirm to complete the G_Write_Gateway_Configuration request to the gateway-internal client.

The session identifier (GS_Session_ID) and transaction identifier (GS_Transaction_ID) are returned to allow matching of the confirm primitive with the original request primitive.

GS_Status is returned to indicate success or failure of the operation, as described in Table U.37.

Table U.37 – GS_Status for G_Write_Gateway_Configuration confirm

Value	Meaning
0	Success
1	Failure; invalid attribute value
2	Failure; other

U.2.4.15 Device configuration interface

U.2.4.15.1 General

The device configuration interface provides a method to manage the configuration of the devices that are associated with a gateway. This is useful for commissioning wireless devices for host systems and related applications.

The device configuration has interfaces to write and to read back the configuration for one or more devices.

A unique identifier is used to match the configuration to a specific device. An IPv6Address is specified for usage in the configuration of the device, allowing subsequent logical access of the device.

The device list report interface is used to determine the devices associated with the gateway. This interface works in conjunction with the device list report interface by providing the ability to limit the devices that are associated with a gateway.

A configuration file may be provided for each device. The format of such a configuration file is gateway-implementation dependent. Information contained in this file is intended to allow gateways to automatically provision devices to join the network. Further configuration is accomplished one-on-one by client server and bulk transfer interfaces.

If the gateway is used to provision devices, the device list report will be empty until devices are provisioned and join the system.

The gateway-internal client uses the G_Write_Device_Configuration primitive to set the configuration for devices associated with a gateway entity.

U.2.4.15.2 Types of primitives and parameters

U.2.4.15.2.1 G_Write_Device_Configuration primitive and its parameters

Table U.38 describes parameter usage for the primitive G_Write_Device_Configuration.

Table U.38 – Primitive G_Write_Device_Configuration parameter usage

Parameter name	G_Write_Device_Configuration	
	Request	Confirm
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Device_List	M	—
GS_Configure	M	—
GS_Unique_Device_ID	M	—
GS_Network_Address	M	—
GS_Provisioning_Info	U	—
GS_Status	—	M

U.2.4.15.2.2 Use of G_Write_Device_Configuration request

The gateway-internal client uses the primitive G_Write_Device_Configuration request to configure the devices associated with a gateway entity.

A session identifier (GS_Session_ID) is obtained from the G_Session interface and included in the request.

A session unique transaction identifier (GS_Transaction_ID) is specified for each invocation of the interface.

A list of devices associated with the gateway entity (GS_Device_List) is supplied. For each device in the list, the unique device identifier (GS_Unique_Device_ID) indicates the device associated with the configuration. If GS_Configure = 1, the configuration is added for the specific device, while if GS_Configure = 0, the configuration is removed for the specific device. A matching IPv6Address (GS_Network_Address) indicates the logical address to associate with the device.

Device provisioning information (GS_Provisioning_Info) is supplied to the gateway for the gateway to control provisioning of the device.

U.2.4.15.2.3 Use of G_Write_Device_Configuration confirm

The gateway entity uses the primitive G_Write_Device_Configuration confirm to complete the G_Write_Device_Configuration request to the gateway-internal client.

The session identifier (GS_Session_ID) and transaction identifier (GS_Transaction_ID) are returned to allow matching of the confirm with the original request.

GS_Status is returned to indicate success or failure of the operation, as described in Table U.39.

Table U.39 – GS_Status for G_Write_Device_Configuration confirm

Value	Meaning
0	Success
1	Failure; invalid or duplicate IPv6Address
2	Failure; out of memory
3	Failure; maximum gateway devices exceeded
4	Failure; provisioning information invalid
5	Failure; other

U.2.4.15.2.4 G_Read_Device_Configuration primitive and its parameters

Table U.40 describes parameter usage for the primitive G_Read_Device_Configuration.

Table U.40 – Primitive G_Read_Device_Configuration parameter usage

Parameter name	G_Read_Device_Configuration	
	Request	Confirm
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Device_List	U	M
GS_Unique_Device_ID	U	M
GS_Network_Address	—	M
GS_Provisioning_Info	—	U
GS_Status	—	M

U.2.4.15.2.5 Use of G_Read_Device_Configuration request

The gateway-internal client uses the primitive G_Read_Device_Configuration request to retrieve the configuration of the devices associated with a gateway entity.

A session identifier (GS_Session_ID) is obtained from the G_Session interface and included in the request.

A session unique transaction identifier (GS_Transaction_ID) is specified for each invocation of the interface.

If a list of devices associated with the gateway entity (GS_Device_List) is supplied, the request is for those specific devices, and the unique device identifier (GS_Unique_Device_ID) indicates the device associated with the configurations to be read. If no list is supplied, the request is for all devices.

U.2.4.15.2.6 Use of G_Read_Device_Configuration confirm

The gateway entity uses the primitive G_Read_Device_Configuration confirm to complete the G_Read_Device_Configuration request to the gateway-internal client.

The session identifier (GS_Session_ID) and transaction identifier (GS_Transaction_ID) are returned to allow matching of the confirm with the original request.

A list of devices associated with the gateway entity (GS_Device_List) is returned. For each device in the list, the unique device identifier (GS_Unique_Device_ID) indicates the device associated with the configuration. A matching IPv6Address (GS_Network_Address) indicates the logical address associated with the device. The device configuration file (GS_Provisioning_Info), when present, provides provisioning information for the device.

GS_Status is returned to indicate success or failure of the operation, as described in Table U.8.

U.3 Example uses of WISN standard services and objects

U.3.1 Tunneling

U.3.1.1 General

The tunnel object (TUN) is a native object that acts as a communication endpoint for the following messaging:

- encapsulated foreign protocol content (shown in Figure U.16 as a dotted line); and
- native interface content (shown in Figure U.16 as a solid line) to configure and manage the tunnel object.

Gateway processes and adapter processes use tunnel objects to support foreign protocol translation. An important aspect of the TUN object is that it provides buffered message behavior for foreign content.

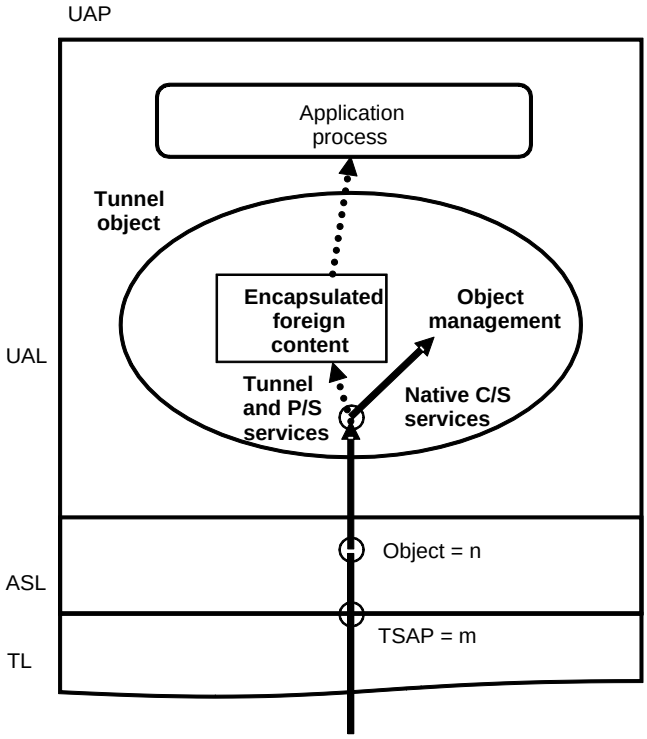


Figure U.16 – Tunnel object model

One or more TUNs may exist within a UAP.

Each TUN object can handle a complete foreign protocol or a portion thereof. Devices that handle multiple foreign protocols will need to implement multiple TUNs. The TUN object is independent of the foreign protocol.

The TUN object relies on the application sublayer (ASL) in order to route messages between peer TUNs and between a TUN object and other non-TUN objects.

U.3.1.2 Distributing tunnel objects

Each device may have one or more TUNs. Tunneling devices includes at least one TUN object as an endpoint for tunnels. Figure U.17 shows a group of related devices with tunnel endpoints interconnected between TUNs.

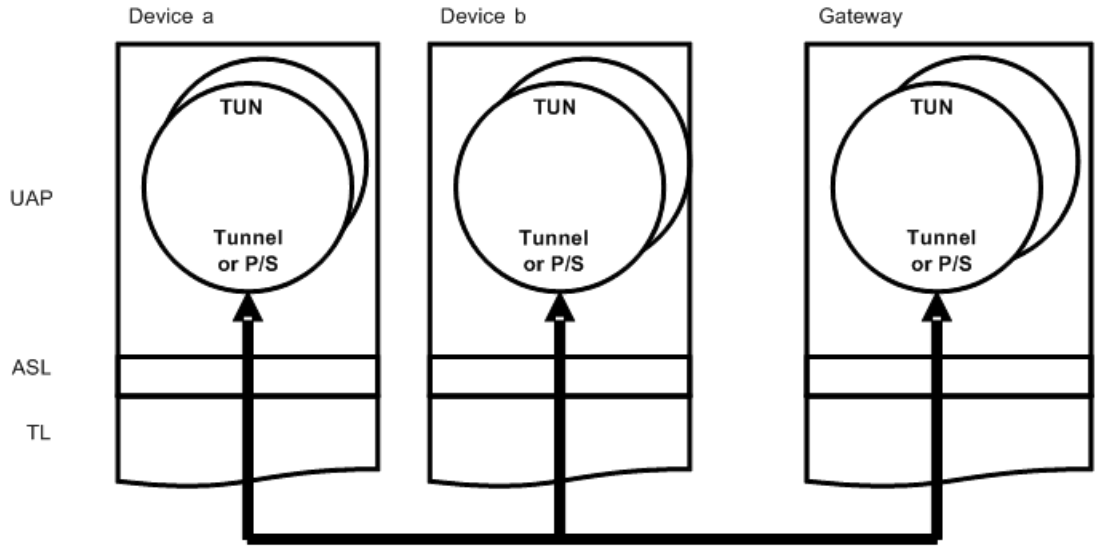


Figure U.17 – Distributed tunnel endpoints

Field devices and adapters may contain TUNs that cooperate with other TUNs in a gateway. A group of related TUNs communicates via a common foreign protocol. The typical usage of TUNs is to communicate between end devices and a host system via the gateway. Direct device-to-device tunneling is also supported within the object model.

TUN object communication is established by using TL interfaces invoked and augmented via the ASL. Communication relationships include publish, subscribe, 2-part tunnel, and 4-part tunnel. Multiple relationships may be established simultaneously.

U.3.1.3 Multicast, broadcast, and one-to-many messaging

As shown in Figure U.18, foreign protocols may require translation of broadcast/multicast messaging relationships when using interfaces such as publish/subscribe and alert distribution. This messaging requires translation support within this standard.

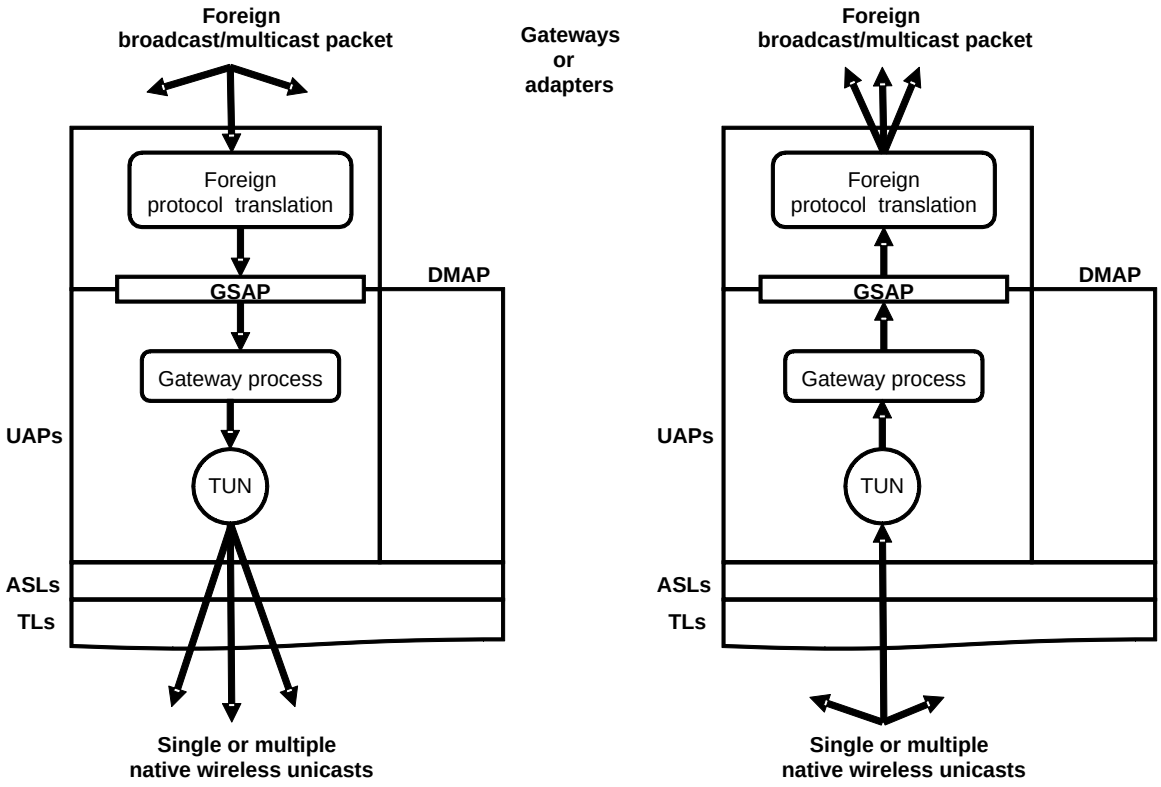


Figure U.18 – Multicast, broadcast, and one-to-many messaging

This standard provides one-to-many messaging support in the tunnel object in order to support translation of the foreign protocol multicast and broadcast requests. The underlying layers of the protocol suite do not provide broadcast or multicast interfaces to the AL. One-to-many messaging is achieved via a series of unicast operations. Protocol translation applications cannot rely on simultaneous delivery of unicast messages.

U.3.1.4 Tunnel buffered message behavior

TUN object communication may be implemented to provide capabilities for buffered and non-buffered behavior for publish/subscribe and tunnel-based message exchanges. TUNs are capable of cooperatively managing buffered behavior to reduce wireless transactions.

NOTE 1 Some legacy protocols use buffered messaging exchanges to support energy-efficient and high-performance protocol translation.

NOTE 2 Some applications are unable to tolerate buffered behavior, usually due to safety and synchronization requirements.

NOTE 3 Buffers are a single element deep. Nothing in this standard prevents implementation of caching and queuing enhancements.

As shown in Figure U.19, each endpoint of a communication flow has a different buffering responsibility, depending on the relationship.

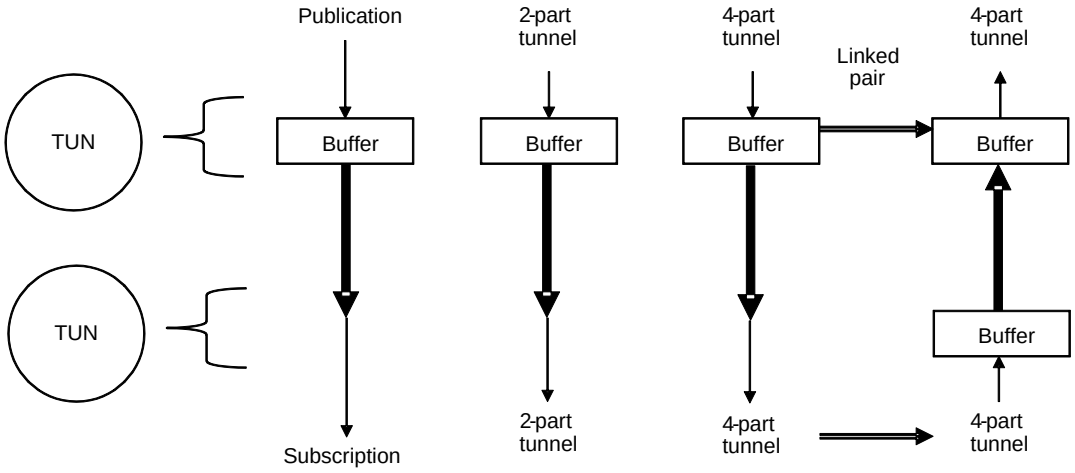


Figure U.19 – Tunnel object buffering

In Figure U.19, two TUN objects are shown. The thin arrows indicate interactions from tunnel applications that use the objects. The thick arrows indicate message flows across the network between the objects. Three types of transaction are shown, a publish/subscribe transaction, a 2-part tunnel transaction, and a 4-part tunnel transaction. Buffers are shown to illustrate the buffered messaging behavior between tunnel endpoints.

A publisher and a subscriber are linked from TUN object to TUN object for periodic updates. Publishers use change of state (CoSt) buffer publications to avoid sending repeated information; subscribers tolerate limited intervals with missing publications.

Messaging with the 2-part tunnel interface behaves in a similar manner, except that the messaging is aperiodic.

Messaging with the 4-part tunnel interface distinguishes a request and a response side via the request/response bit in the tunnel interface header. The request side buffers the first request and also forwards the request. A response is generated as indicated by the double arrow on the response side in Figure U.19. The response is stored in a second linked buffer on the request side, as indicated by the double arrow in Figure U.19. Change of state processing applies to subsequent duplicate requests, wherein the response is returned from the local buffer. Where change of state indicates an altered request, the request is forwarded and the local response buffer is updated. A final buffer is shown in Figure U.19 on the response side. This buffer supports change of state behavior wherein a single request results in multiple responses over time to update the request side.

U.3.1.5 Tunnel object attributes

TUN object attributes are described in Clause 12, but are further described herein.

The Protocol attribute is used to configure the protocol associated with the tunnel object and the associated remote tunnel objects. When the protocol is set to none, the tunnel can be configured. Once another protocol is set, the tunnel object configuration is applied and the status is updated to reflect the result.

The tunnel endpoint structure describes address information pointing to a single remote tunnel object. The array of tunnel endpoints allows specification of one or more tunnel endpoints representing remote tunnel objects. This allows a single communication relationship to span multiple tunnel objects where necessary. Max_Peer_tunnels indicates the maximum number of entries in the array. Num_Peer_Tunnels indicates the actual number of entries configured in the array.

One of several types of communication flow types is selected between the tunnel objects by configuration of the Flow_Type attribute. Flow types include 2-part tunnel, 4-part tunnel, publish, and subscribe.

For publish and subscribe Flow_Types, the Update_Policy allows configuration of periodic publication or change of state publication. Periodic publication occurs at every opportunity. CoSt publication occurs only when fresh publication data is available. The publication frequency is based on the Period attribute. The actual timing is based on a combination of the Period and the Phase attributes. The Stale_Limit is used in the subscriber to configure behavior for detection of excessive publication loss or delay. Stale_Limit is a multiplier that configures the number of periods that a subscriber will wait before considering lost publications to indicate a problem.

Foreign_Destination_Address and Foreign_Source_Address are the addresses associated with the tunnel endpoint by the foreign protocol. The format is dependent on the foreign protocol. These addresses are returned to protocol translator applications as tunnel object messages are received. They allow utilization of IPv6Addressing as defined in this standard in lieu of carrying the foreign addressing. Mapping via the tunnel object allows reconstruction of foreign PDUs containing address information.

NOTE Depending on the specific foreign protocol conversion, the foreign PDU will vary. Most fieldbus protocols will form DPDUs for direct delivery on a local link. In contrast, IP-based protocols usually form NPDUs, where a final encapsulation is achieved by an address resolution protocol.

Connection_Info[ ] and Transaction_Info[ ] are octet strings that are written by the protocol translator as required. Connection_Info[ ] is used to provide protocol specific static message content on message receipt in order to eliminate the repeated wireless message transfer of the content. Transaction_Info[ ] is used to provide protocol specific message content on receipt of a response, where the content would otherwise be echoed from the request in the response, eliminating the wireless transfer of the content. Further description is provided in U.3.1.9 and Annex O.

It is the responsibility of the TUN object implementation to maintain a related contract for each tunnel endpoint.

U.3.1.6 Tunnel object messaging

U.3.1.6.1 Application sublayer interface usage

TUN objects may be implemented to provide connection interfaces that include a publish/subscribe interface, a 2-part tunnel interface, and a 4-part tunnel interface. Each interface may be implemented in both a buffered and a non-buffered mode of operation.

An optional external interface for invoking the gateway connection interfaces is described in Clause U.2.

The TUN object uses the ASL to deliver and receive interface content as described for the publish interface in 12.17.3.2 and the tunnel interface in 12.17.6.2. The ASL provides object-to-object delivery of publish/subscribe payloads in external formats through the publish request primitive. The ASL also provides a linked tunnel request and tunnel response primitive.

The header is described in 12.22.2.3. This header enables request and response specification, interface type specification (publish or tunnel), and object identifier addressing mode (4-bit, 8-bit, or 16-bit). A large number of tunnel objects will result in a larger address space and more overhead in the header.

The publish interface payload format is described in 12.22.2.12 by Table 348. There is no explicit size in the header. The size of the publication is supplied with the publish request and is known to the subscriber by information supplied with the indication.

The tunnel interface request and response payload formats are described in 12.22.2.9. The request allows 7-bit size (0..127 octet payloads) or 15-bit size (128..32 767 octet payloads). Inclusion of the size allows tunnel message to be concatenated by the ASL.

NOTE Most encapsulated messages from legacy protocols referenced by this standard fall into the range of less than a 127-octet payload, resulting in a 7-bit field.

U.3.1.6.2 Information classification and transfer rules

From a caching and buffering viewpoint, information may be classified as constant, static, dynamic, or non-cacheable. These classifications are described in 12.6.3 for native object attributes. The same guidance applies to the selection of buffering for publish and tunnel interfaces for foreign payloads.

Constant information should not be transferred more than once between TUNs, except where local copies are lost due to power cycling, reset, cache deletion, or elimination of references to the information.

Static information should not be transferred more than once between TUNs, except as indicated for constant information and where the static information has been modified.

Dynamic information should only be transferred between TUNs when its value has changed unless it is required more often to indicate that the source or destination is still active.

Non-cacheable information may be transferred between TUNs on each request.

U.3.1.6.3 Publish/subscribe interface

The tunnel object may be implemented to provide facilities to accomplish buffered and non-buffered publish/subscribe messaging for dynamic information update.

The flowcharts of Figure U.20, Figure U.21 and Figure U.22 describe the behavior of TUN object publishers and subscribers that use buffering. The behavior describes the base message transfer agreement between a publisher and a subscriber based on the TUN object attribute configurations.

NOTE The interpretation and actions for initial, stale, and repeat data are based on implementation, as is the CoSt algorithm.

The publish/subscribe publisher connection operates as shown in Figure U.20 when CoSt updates are configured.

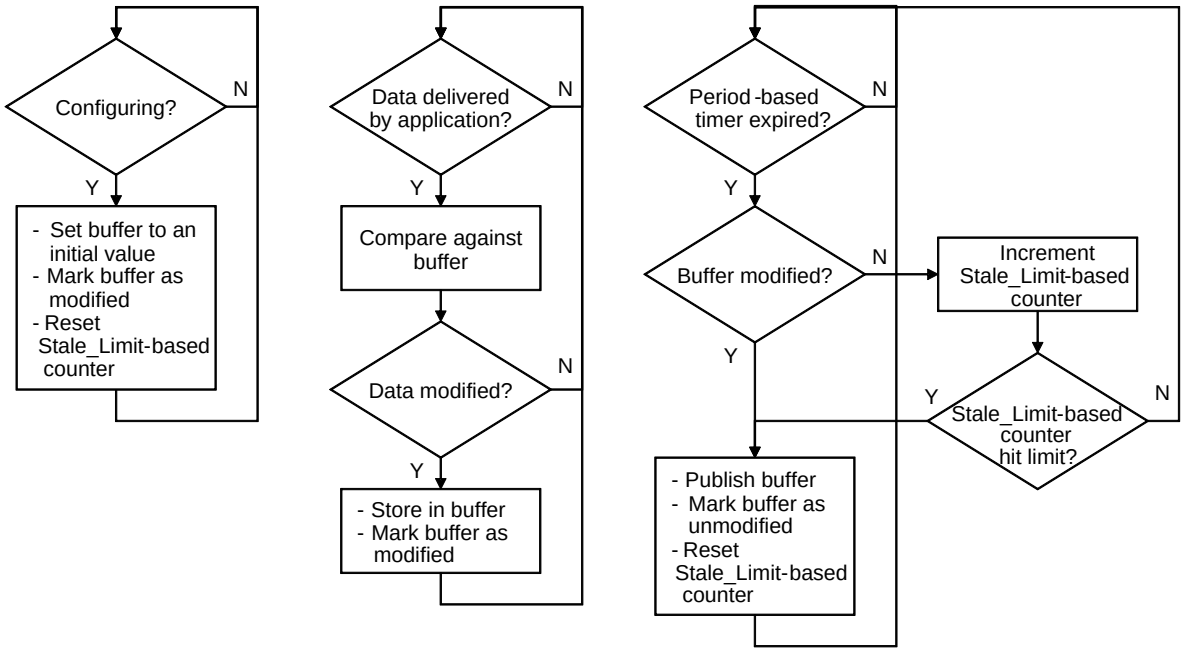


Figure U.20 – Publish/subscribe publisher CoSt flowchart

The publish/subscribe publisher connection operates as shown in Figure U.21 when periodic updates are configured.

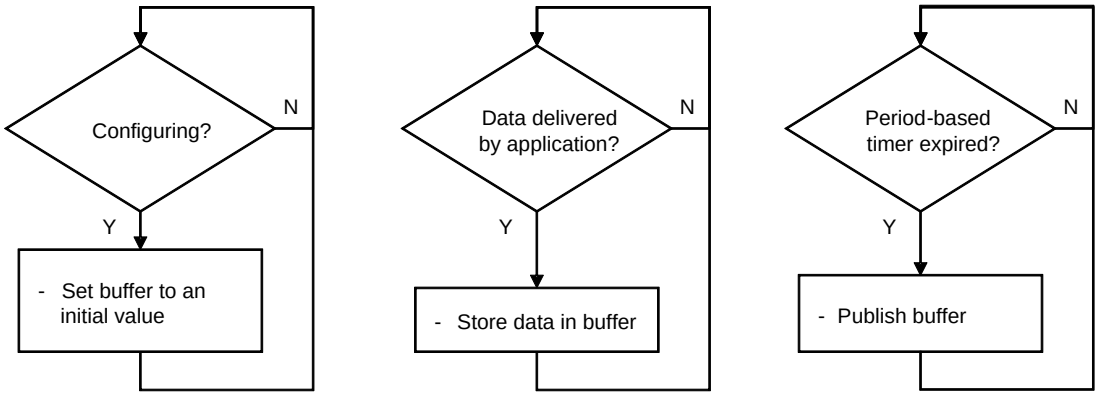


Figure U.21 – Publish/subscribe publisher periodic flowchart

The publish/subscribe subscriber connection operates as shown in Figure U.22 when periodic or CoSt updates are configured.

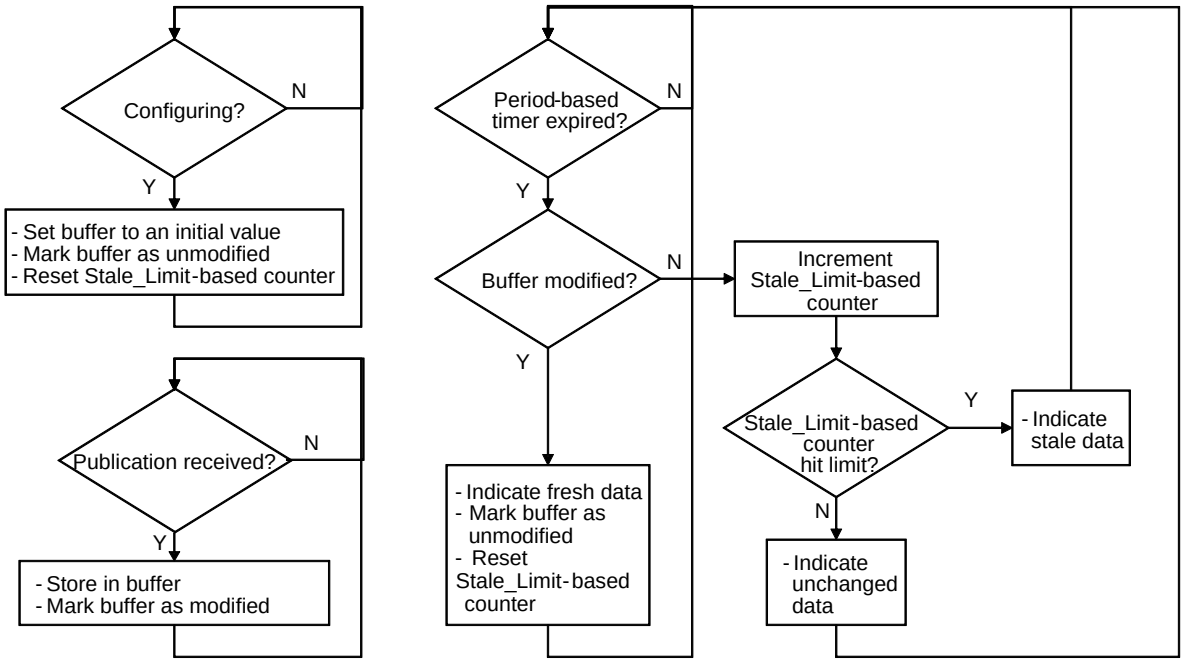


Figure U.22 – Publish/subscribe subscriber common periodic and CoSt flowchart

U.3.1.6.4 Tunnel interface

The tunnel object may be implemented to provide facilities to accomplish buffered and non-buffered tunnel interface messaging. Non-buffered tunnel interface messaging provides support for unconditional transfer of non-cacheable and constant information. Buffered tunnel interface messaging provides support for buffering and contingent transfer of static and dynamic information.

U.3.1.7 Multiple server responses

Certain client/server requests receive multiple responses. One reason is that the request requires extended processing and an immediate response is sent which indicates that the request was received and that the real response will be sent after processing is complete. This is known in some protocols as a delayed response. In other cases, the server provides additional updates over time to satisfy the initial request. Certain protocols collect process variables or historian information in this manner.

The client/server buffered and unbuffered interfaces support multiple application responses for these purposes. In the case of the buffered response, the read buffer maintains the latest response. The client receives an indication on each response.

U.3.1.8 Tunnel object address mapping

The TUN object may be implemented to contain three address fields (Foreign_Destination_Address, Foreign_Source_Address, and the Array of Tunnel endpoints) that are used in the translation between foreign addresses and native addresses.

As shown in Figure U.23, foreign multicast and foreign broadcast addresses require translation to native addresses and messaging.

The first case is where the multicast or broadcast originates on the foreign network. Since multiple hosts, protocols, or applications may share a wireless network as described herein, sending a foreign broadcast to all wireless devices is inefficient. Thus, foreign broadcast into the wireless network uses simulated multicast to a limited group. The TUN object is used to

simulate multicast delivery (one-to-many messaging) by maintaining a list of unicast addresses (array of tunnel endpoints) and by using a sequence of unicast deliveries.

The second case is where the multicast or broadcast originates on the wireless network and is destined for the foreign network. A single APDU is delivered from the wireless network and acted on by a protocol translator to generate a multicast or broadcast PDU on the foreign network.

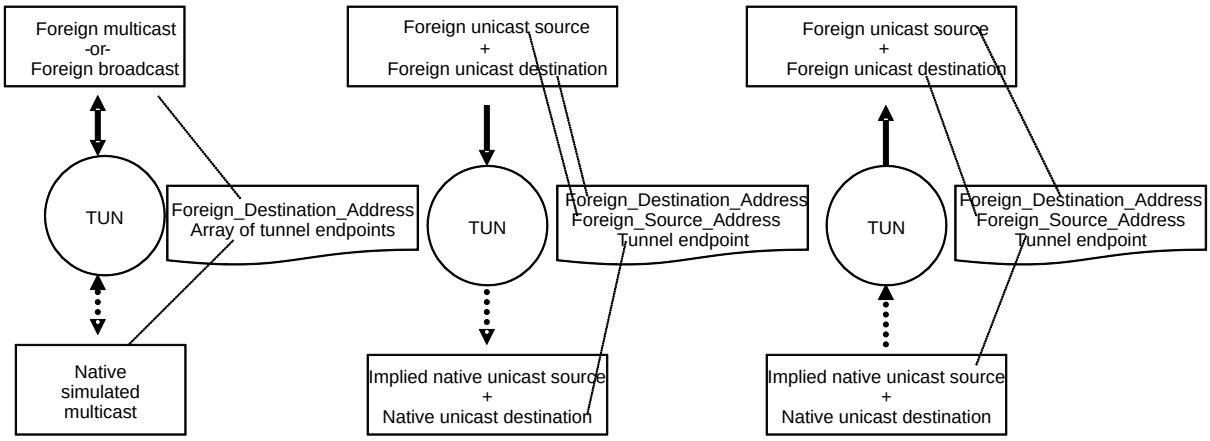


Figure U.23 – Network address mappings

Also shown in Figure U.23 is a pair of unicast address translations.

The first case translates from a foreign source/destination address pair to a native address pair. The second case translates from a native source/destination address pair to a foreign address pair. Both the Foreign_Destination_Address and the Foreign_Source_Address are used. Only the address information from a single tunnel endpoint is necessary, since the TUN object has access to its own native address for usage in source or destination fields. Foreign source and destination definition depend on the direction of the transfer.

U.3.1.9 Connection and transaction information

TUN objects function as initiator endpoints (publisher and tunnel request) and correspondent endpoints (subscriber and tunnel response). Protocol translation sends foreign content as TUN-DATA between the endpoints. Since most legacy protocols are not optimized for low-energy wireless communication, various mechanisms are available to increase efficiency.

When a protocol translator tunnels a foreign PDU, it is not efficient to repeatedly send static portions of the foreign PDU between the endpoints. Such static information includes preambles and secondary fixed addressing, such as logical unit identifiers. As shown in Figure U.24, TUN objects provide a generic mechanism (Connection_Info) for provision of static information on foreign PDU receipt without per-message wireless transfer of the static information.

NOTE Depending on the specific foreign protocol conversion, the foreign PDU will vary. Most fieldbus protocols will form DPDUs for direct delivery on a local link. In contrast, IP-based protocols usually form NPDUs, where a final encapsulation is achieved by an address resolution protocol.

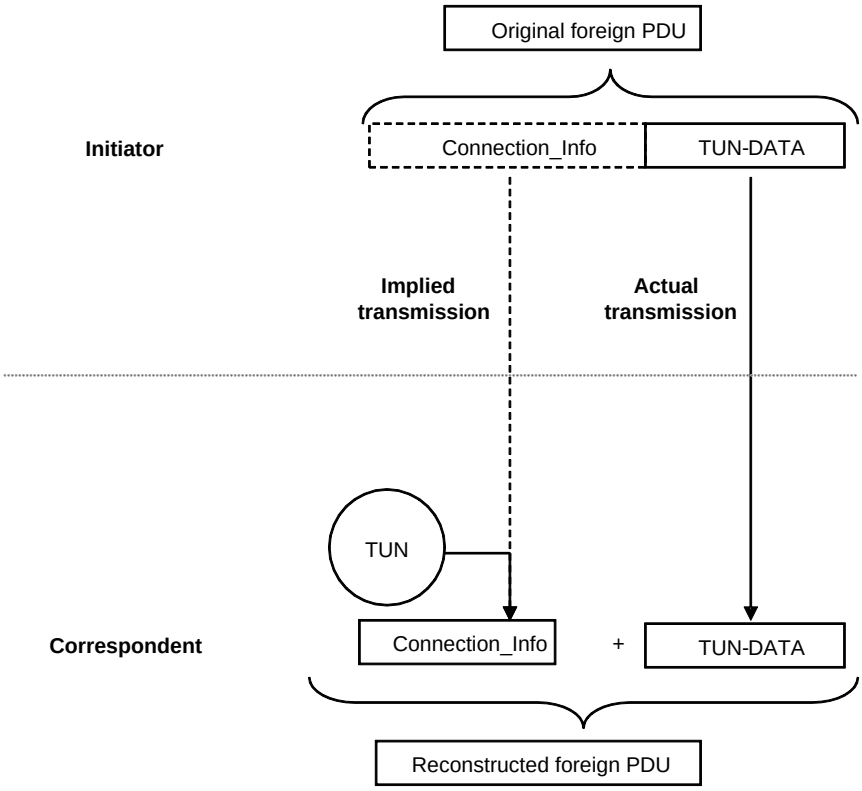


Figure U.24 – Connection_Info usage in protocol translation

When a protocol translator performs a transaction, it is not efficient to carry transaction-specific information that is only used to identify the transaction at the initiator. Such information includes information to link the original request to the response, where knowledge of the endpoint can be used. As shown in Figure U.25, TUN objects provide a generic mechanism (Transaction_Info) for provision of transaction-specific information without carrying the overhead in the wireless transfer.

Both Connection_Info and Transaction_Info can be used simultaneously.

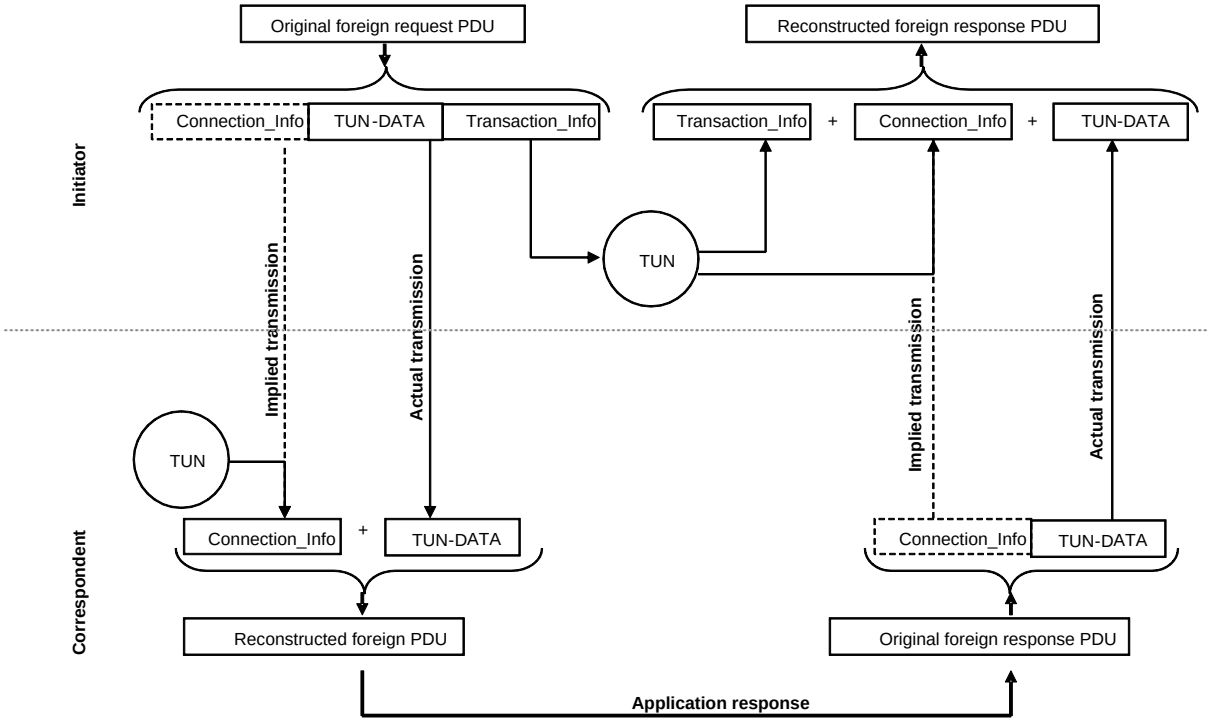


Figure U.25 – Transaction_Info usage in protocol translation

U.3.1.10 Interworkable tunneling mechanism

U.3.1.10.1 Overview

Annex U describes a communication mechanism for foreign network nodes to communicate across a wireless network via gateways and adapters. This mechanism enables vendor-independent development of interworkable gateways and adapters by implementing a restricted subset of the communication features defined within this standard. The interworking communication is achieved by the use of a constrained tunneling mechanism. The gateways and adapters serve to interconnect two or more foreign network segments by bridging foreign protocol DPDUs through the wireless network as depicted in Figure U.26. The gateway and adapter application processes use the AL tunnel objects and interfaces for the exchange of foreign unicast DPDUs and foreign broadcast/multicast DPDUs. The mechanism by which the gateway and adapter application processes exchange these DPDUs with foreign nodes is not specified by this standard.

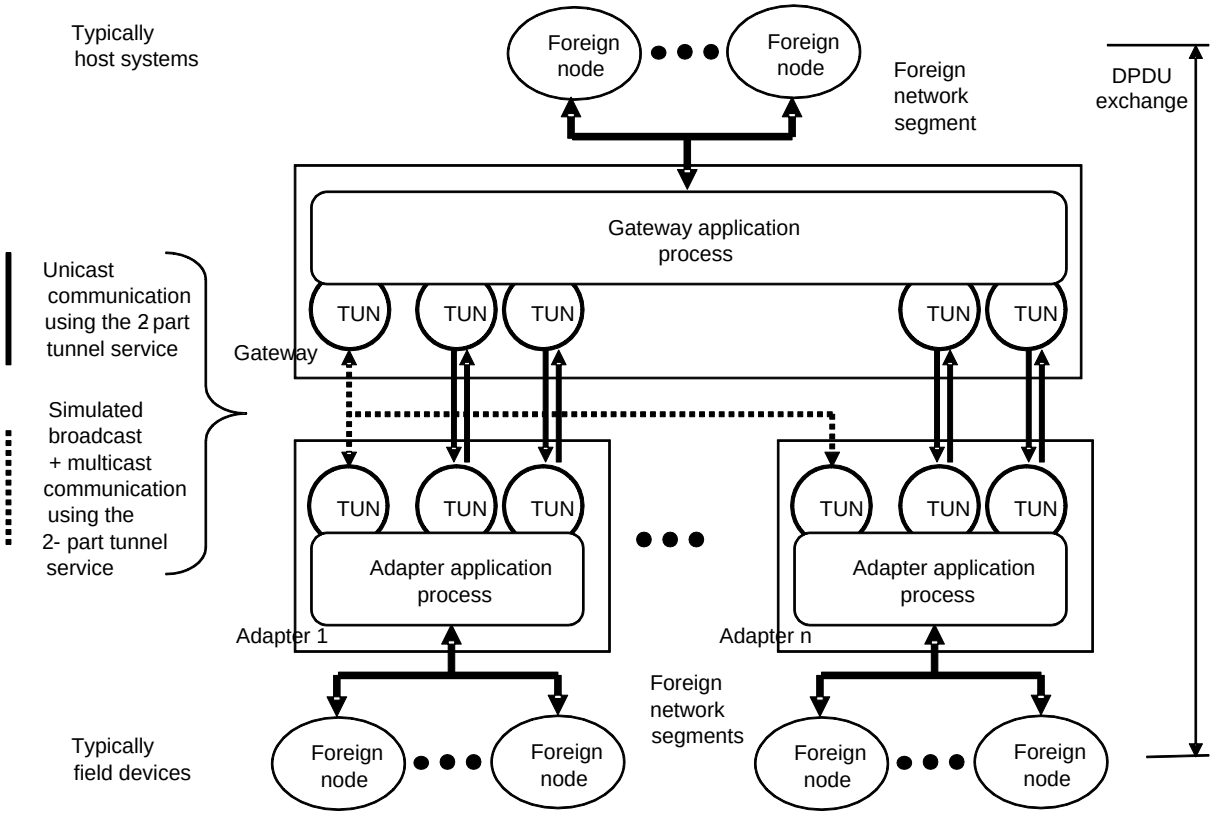


Figure U.26 – Interworkable tunneling mechanism overview diagram

U.3.1.10.2 Tunnel object placement

One or more foreign network nodes, individually addressable by a unicast DPDU address, may exist behind a gateway or an adapter. A foreign network node behind a gateway or adapter may require communication with an associated foreign network node behind another gateway or adapter.

For each associated gateway and adapter, a tunnel object is disposed and configured to carry foreign broadcast and multicast addressed DPDUs, one for a first associated foreign network segment and one for each additional associated foreign network segment.

For each associated foreign network node pair, a pair of tunnel objects is disposed and configured to carry unicast addressed DPDUs, one in the gateway or adapter for a first associated node and one in the gateway or adapter for a second associated node.

U.3.1.10.3 Tunnel object configuration

Tunnel operation is controlled as described in Clause 12. Tunnel objects are configured through attribute settings. Changes to the configuration are required to be correctly sequenced by setting the Protocol attribute and monitoring the Status attribute.

The unicast tunnel object pairs are configured as follows:

- The Flow_Type attribute is configured for a 2-part tunnel.
- The Array of Tunnel endpoints attributes are configured for a single address element, where each tunnel object in the pair addresses the other tunnel object in the pair.
- The Connection_Info[ ] and Transaction_Info[ ] attributes are not used.
- The Update_Policy, Phase, Period and Stale_Limit attributes are not used.

For unicast tunnel objects, the Foreign_Destination_Address attribute of each local tunnel object is set to the DPDU address of the associated foreign device behind the remote gateway or the adapter and the Foreign_Source_Address attribute of each local tunnel object are set to the DPDU address of the associated foreign device behind the local gateway or adapter.

The tunnel objects in the broadcast/multicast tunnel object set are configured as follows:

- The Flow_Type attribute is configured for a 2-part tunnel.
- The Array of Tunnel endpoints attributes are configured for one or more address elements, where each tunnel object in the set addresses all other tunnel objects in the set.
- The Connection_Info[ ] and Transaction_Info[ ] attributes are not used.
- The Update_Policy, Phase, Period and Stale_Limit attributes are not used.

For broadcast/multicast tunnel objects, the Foreign_Destination_Address attribute and the Foreign_Source_Address attribute are set to an equal value.

The usage of the Foreign_Source_Address attribute and the Foreign_Destination_Address attribute enables gateways and adapters using the interworkable tunneling mechanism to be configured strictly by configuration of the tunnel objects.

Associated gateways and adapters may send and receive foreign DPDUs from either identical versions or interworkable versions of the same foreign protocol. To enable the run state after the other attributes are configured, the Protocol attribute is configured last and is configured to the same value in all tunnel objects associated with all related foreign network segments on the D-subnet. The final Protocol attribute value is set as defined in Annex K. The gateway and adapter application processes may report tunnel object Status = 2 (configuration failed) if an attempt is made to configure a tunnel with an unsupported Protocol.

A compatible foreign protocol may be able to accommodate the timing imposed by the wireless mechanisms, either inherently or by configuration. Exchange of foreign DPDUs may not be the most efficient tunnel method, but it assures that sufficient information is available to process the packet within gateway and adapter application processes. It also assures multiple vendors convey the same information between gateway and adapter application processes. It is also assures that sufficient information is available within gateway and adapter application process to link client/server requests and responses. Addressing is also carried and enables multiple foreign network devices to sit behind each gateway or adapter.

U.3.1.10.4 Tunnel operation

Foreign network DPDUs may be delivered to the gateway and adapter application processes in one of two ways, either through a tunnel object or from a foreign source outside of the wireless network. The outside source will usually be a wired network (and its associated protocol stack) attached directly or indirectly to the gateway or adapter. Alternatively, the PDUs may be generated by software or firmware interacting with (or embedded in) the gateway or adapter application process directly. In either case, the tunneled PDU exchange between gateways and adapters may remain identical.

The gateway and adapter application processes may examine the foreign network DPDU destination address prior to forwarding the PDU over the wireless network. DPDUs without a known destination that is reachable through the tunnels are not forwarded.

Gateways and adapters application processes may forward foreign protocol unicast DPDUs to DPDU address destinations that are reachable through a linked pair of unicast tunnel objects.

Gateway and adapter application processes forward valid foreign protocol broadcast and multicast PDUs through the broadcast/multicast tunnel that exists within each associated

gateway and adapter, distributing the same PDU to one or more destinations. The PDU is not echoed back to the source.

The gateway and adapters application processes may use multicast group establishment PDUs from within the foreign protocol, where such PDUs exist, in order to limit the distribution scope.

Since generation of multiple copies of the same message is almost certain to occur, the foreign protocol may tolerate timing skew.

U.3.1.10.5 Efficient operation

It is recommended that foreign protocols that are using the interworkable tunneling mechanism reduce PDU exchanges to the minimum that is acceptable to the foreign protocol and its applications. This is accomplished by extending update periods and timeouts for periodic update. This is also accomplished by elimination of redundant transfer of static information by maintaining local copies.

U.3.2 Bulk transfer

Large item transfer is accomplished through upload/download objects (UDOs), as shown in Figure U.27. Large item transfers are useful for firmware updates, transfer of large sample buffers such as captured waveforms, and general configuration. One UDO represents a single item that can be transferred in either direction (uploaded or downloaded) to/from another application. The item to be transferred exists at the location of the UDO. Interface objects (IFOs) act as clients to initiate transfers. The transfer protocol provides buffering, flow control, and guaranteed and in-order delivery. Protocol translators have access to the UDO through the GIAP.

Items are associated with a string that can be used to encode item specific identification and revision information. Asset management systems can be constructed to monitor revisions for regulated industries and to backup and restore items generically, without knowledge of the item content. Protocol translators may also transfer large items via foreign protocols through tunneling, but this precludes protocol independent asset management.

End applications are expected to understand the content of the transferred item and how to apply it. Provisions exist (depending on device capabilities) to request utilization of the item (possibly altering run-time behavior) and for storage of the item in non-volatile memory.

The UDO and the upload and download bulk transfer protocol are described in 12.15.2.4.

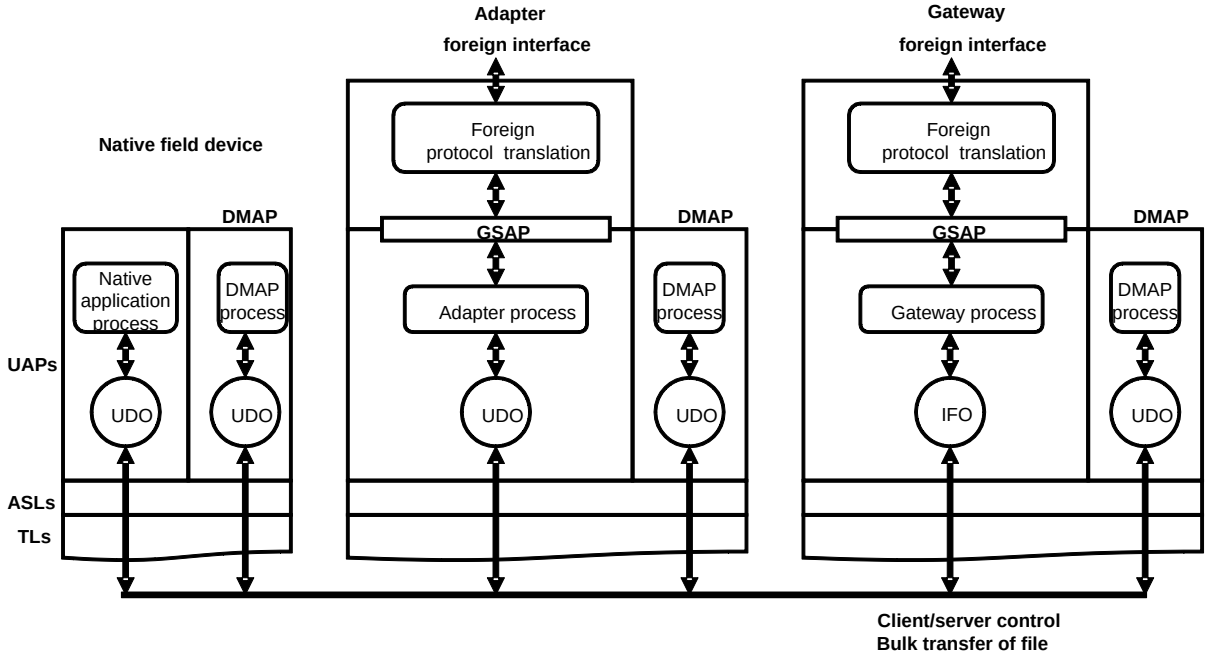


Figure U.27 – Bulk transfer model

U.3.3 Alerts

Alerts may be generated by many of the objects defined by this standard. Some objects reside within device UAPs, while others reside in the DMAP as management objects for each layer.

Alerts within a device are consolidated within the alert reporting management object (ARMO). Each device has a single ARMO that resides within the DMAP. All alerts within a device are conveniently consolidated in this single location.

The ARMO in each DMAP is responsible for reporting alerts through an AlertReport interface to an alert-receiving object (ARO). The ARO acknowledges alert receipt through the AlertAcknowledge interface. This transfer occurs independently of the actual processing of the alerts.

Alerts fall into four categories:

- process;
- device;
- network; and
- security.

Each category can be delivered by an ARMO to a different ARO. Thus, a single ARO might collect all process alerts across an entire network, or a set of AROs can be used, with each ARO only collecting a single category of alerts. If each ARO collects only one type of alert, then collection of all alerts requires four AROs.

The gateway contains one or more AROs that allow collection, reporting, and management of alerts.

Protocol translators have access to and can manage alerts through the GIAP, as shown in the alert model in Figure U.28. Subscription interfaces allow alert selection through the GIAP.

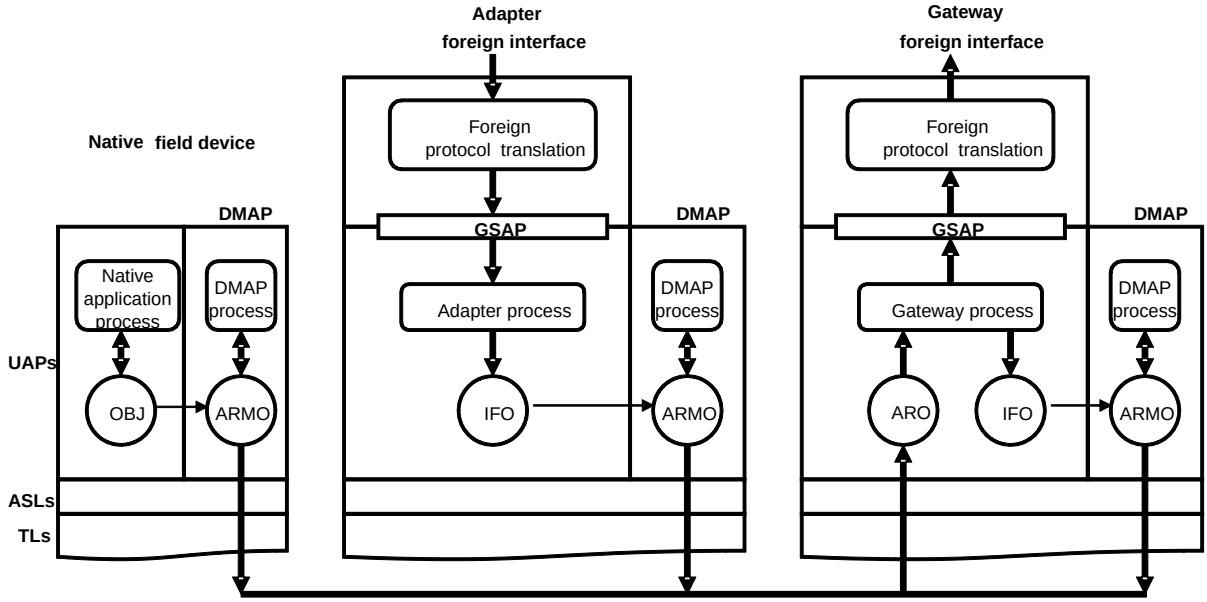


Figure U.28 – Alert model

Alerts fall into two classes, alarms and events. Events are informational and generate event messages through the GIAP. Alarms have states and require alarm-specific actions to clear the alarms. Usually, client/server messaging is used to perform these actions.

The gateway and the adapter applications are also able to generate native alerts from IFO instances. This allows protocol translators to generate alerts within the context of a standard alert management system.

In certain circumstances, the state of alerts may be lost at the ARO, such as when a gateway is reset or replaced. In such case, the original ARMOs will no longer contain information about events, but will maintain state information related to alarms. An alarm recovery procedure can be initiated in order to recover the system alarm state.

This standard does not support multicast alerts. As a result, the same alerts cannot be routed to both the gateway and the system manager if they are not physically co-located. Network and security alerts are currently sent to the system manager by default. Process and device alerts may be sent to the gateway role.

The alert model does not support multicast alerts. Network alerts and security alerts are potentially useful in a gateway for transformation into generic foreign protocol error messages. The system manager is the default destination for these alerts. In system configurations where the system manager is connected to the WISN via the gateway, the ARO in the gateway, when configured to collect alerts for network and security purposes, is capable of reposting the alerts through the local ARMO to the system manager. This is illustrated in Figure U.29.

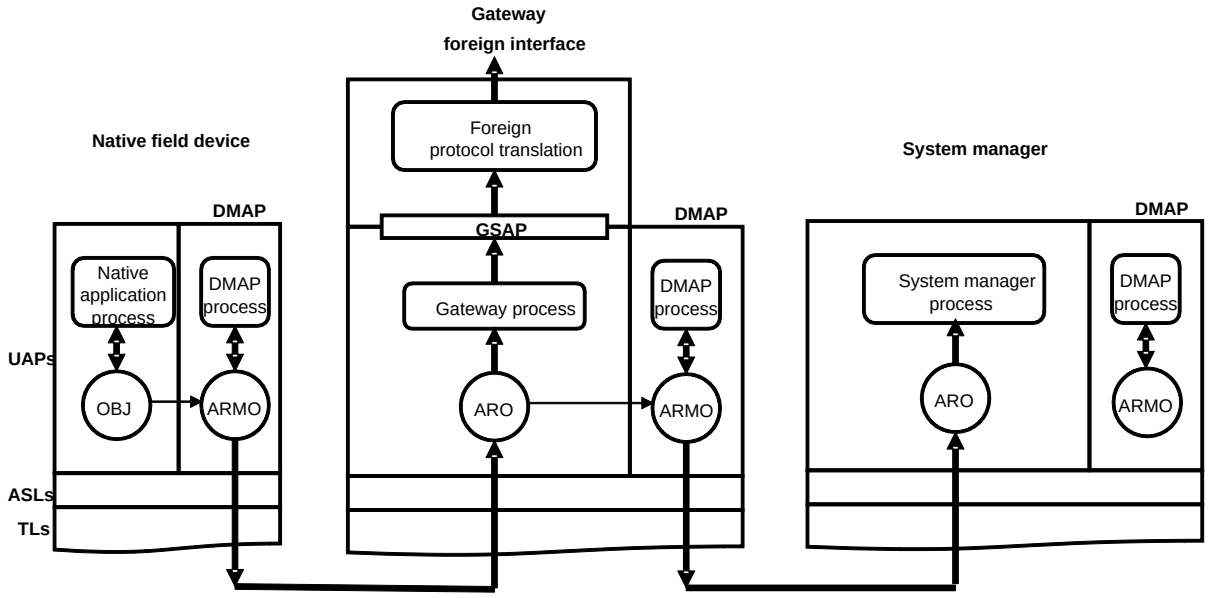


Figure U.29 – Alert cascading

U.3.4 Native publish/subscribe and client/server access

This standard provides publish/subscribe and client/server interfaces via the ASL that is used to interact with application-specific native objects.

For publish/subscribe, the concatenation (CON) and dispersion (DIS) objects are used as endpoints.

For client/server interfaces, two object endpoints are required in order to use these interfaces. The IFO may act as one endpoint for these interfaces within gateways. Any other application or management object within the system can act as the other endpoint.

As shown in Figure U.30, utilization of these objects allows protocol translators to integrate simple devices that do not include legacy protocols.

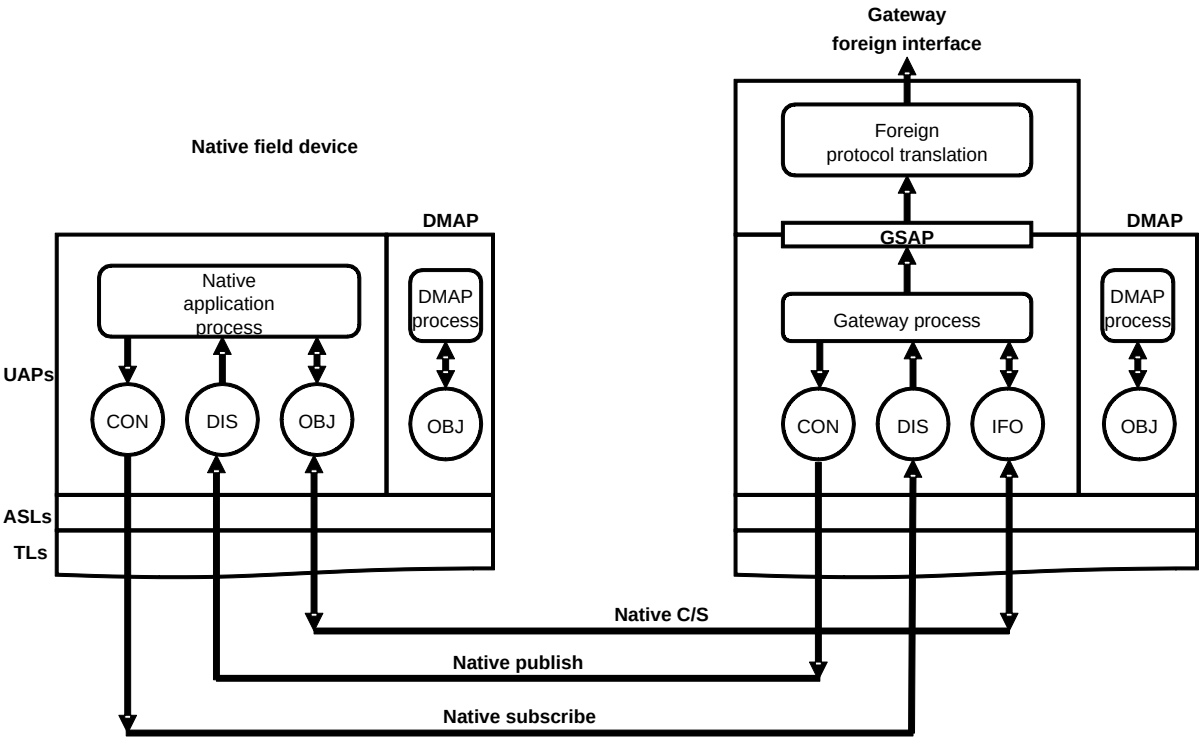


Figure U.30 – Native publish/subscribe and client/server access

Within a gateway, the CON and DIS objects may provide buffered message behavior for change of state operation.

Within a gateway, the IFO may provide buffered message behavior as described for the 4-part tunnel messaging between tunnel objects for client/server read interfaces. The IFO may use the attribute classification to determine buffering behavior. Non-cacheable attributes are not buffered. Constant attributes are buffered. Static and dynamic attribute buffering is determined by application requirements.

The protocol translator may use native addressing (Network_Address, Transport_Port, OID, and attribute identifier) to identify native messages.

NOTE Tunneling assumes that foreign protocol messages are transferred between endpoints. As such, foreign addresses are associated with the messages and used for teardown and reconstruction of the messages in order to avoid transfer. No such assumption is made for native messaging, where a one-to-one message flow is less likely to exist.

U.3.5 Time management

Host time may be propagated through a gateway to a wireless system, giving the host system and the field devices the same sense of time (within tolerances). This enables the host time to be used for purposes such as uniform alert timestamping and sequence of event determination that spans wireless and wired devices connected to the host. Without periodic synchronization to host time, the wireless system will drift, thus periodic adjustment capability is desirable. Both the host and wireless system may be synchronized to a common external source such as a GPS derived timesource.

To propagate host time, a gateway may perform periodic synchronization of time in an attached D-subnet time to an external source by requesting time changes through a DLMO.

Protocol translators within a gateway may access time management functions through the GIAP services. Protocol translators are responsible for accessing external time sources and

converting protocols and time formats. Network time is represented in TAI format, as described in 5.6.

A DL configured as a clock master is used to propagate time synchronization information to an attached D-subnet, as described in 6.3.10.3. Each node contains a DMO within its DMAP. The DMO contains attributes DL_Subnet_Clock_Master_Role and DL_Subnet_Clock_Repeater_Role that control the ability of a node to be a clock master. Allocation of the clock master role is coordinated with the system manager. The device registers its ability to be a time source during the subnet joining process.

The DLMO contains an attribute called TaiTime that reports the current time and another attribute called TaiAdjust for adjusting the time. The DLMO is used to adjust the time of the D-subnet.

One or more DLs may be associated with a gateway. In one implementation, the DLs are integrated within the gateway. In another implementation, the DLs are within backbone routers and separated from the gateway, adding indeterminate delays. Each implementation may consider the implications of delay associated with access of DL objects to perform synchronization.

U.3.6 Security

Sets of wireless devices are related to a foreign host via a gateway. The gateway and the wireless devices are expected to belong to a common security group. Security for this group may be established by MAC or TL security configuration, or both as described in Clause 7. Establishment of common security settings is a prerequisite for communication between protocol translation communication endpoints.

Common foreign fieldbus protocols do not have security capabilities. This does not preclude extension of secured protocols into this standard's domain. It is the responsibility of foreign protocol translators in both gateways and adapters to act as trusted applications in the extension of foreign protocol security from end-to-end. This can be achieved by utilization of native security or through tunneled exchanges.

U.3.7 Configuration

For gateways which implement internal interfaces such as the example GSAP interfaces, it is recommended that the gateway entity configuration/capability be made internally available to gateway internal clients. How these gateway internal operational attributes are made available is a local matter. Examples attributes which the gateway GSAP entity may wish to present are described in Table U.41. For convenience the attribute describing conventions used are those used by other clauses of this standard. See also the G_Read_Gateway_Configuration interface example for an example of how an interface can be used to make this information available to a gateway-internal client.

Table U.41 – Example of gateway configuration management attributes

Standard object type name: not applicable				
Standard object type identifier: not applicable				
Attribute name	**Attribute identifier**	**Attribute description**	**Attribute data information**	**Description of attributes behavior**
Max_Devices	11	Maximum number of devices supported by gateway	Type: Unsigned16	Implementation dependent value set by gateway depending on resources
			Classification: Static	
			Accessibility: Read only	
			Default value: 0	
Actual_Devices	12	Current number of devices connected to gateway	Type: Unsigned16	Increases and decreases based on devices in communication with the gateway
			Classification: Dynamic	
			Accessibility: Read only	
Max_Leases	13	Maximum number of leases supported by the gateway	Type: Unsigned16	Implementation dependent value set by gateway depending on resources
			Classification: Static	
			Accessibility: Read only	
Actual_Leases	14	Current number of leases for devices connected to the gateway	Type: Unsigned16	Increases and decreases based on leases. Device complexity will determine the number of leases required
			Classification: Dynamic	
			Accessibility: Read only	

U.3.8 Provisioning and joining

A gateway is a network device as described in this standard and is provisioned using the generic methods described in this standard.

A gateway that communicates to D-subnets through backbone routers may provide a method to configure the gateway to communicate to a specific D-subnet and to specific devices within that D-subnet through a specific backbone router.

NOTE 1 Nothing precludes more dynamic implementations, such as a load-sharing algorithm that assigns devices to the best BBR found, or gateways and BBRs that discover each other, or support for redundancy that is provided automatically where D-subnets overlap.

A gateway that communicates to one or more D-subnets through backbone routers includes a method to configure the gateway to communicate with at least one system manager, where the system manager may reside:

- in the gateway;
- on a backbone that the gateway can use for communication; or
- within a D-subnet that the gateway can use for communication.

A gateway is a network device (containing an AL and IPv6Address) as described in this standard and joins the network following the join methods described in this standard.

NOTE 2 Several variations are possible, for example: a gateway that joins by sending an internal request to a co-resident system manager, or by sending a join request through a local PhL, or that uses a backbone router's PhL indirectly, or that sends the join request across the backbone to a system manager.

Annex V
(informative)

Compliance with ETSI EN 300 328 v1.8.1

NOTE 1 With slight adjustment (i.e., judicious addition of a few “shall”s), this Annex V could be made the normative basis for operation in the EC. In that case ETSI EN 300 328 would need to be promoted from the Bibliography to the normative references of Clause 2.

ETSI EN 300 328 is a complexly-interacting set of requirements which mandates specific declarations of operating behavior. It appears that there are several different operating modes under which a device conforming to this IEC standard can comply with those requirements, which are generally more favorable for operation of this IEC standard than those of other ETSI ENs such as ETSI EN 300 440.

Devices that comply with ETSI EN 300 328 are permitted to change their operating mode dynamically. Presumably each potential operating mode would need to be assessed independently for conformance.

Under ETSI EN 300 328, IEEE 802.15.4 2,4 GHz DSSS qualifies as wideband modulation (WBM). As used in this IEC standard it also qualifies as frequency-hopping spread-spectrum modulation (FHSSM) whenever the cyclic frequency-hopping schedule specifies at least 15 channels.

NOTE 2 Even with WBM, some frequency hopping is needed to avoid commonly-encountered narrow-band fading with a duration of more than a few ms. Thus frequency hopping will occur whether it is claimed for operation under FHSSM mode relative to conformance to ETSI EN 300 328, or not.

NOTE 3 ETSI EN 300 328:2012 v1.8.1, 4.3.1.3.2 permits blocking transmissions on some of the channels specified in the frequency-hopping schedule (black listing), but does not permit the number of different hopping channels to be reduced to fewer than 15 channels. Therefore inclusion of fewer than 15 channels in a channel map that determines the frequency-hopping cycle of nominally-active channels means that the only possible remaining operating regimes are those under WBM.

In this IEC standard, D-transaction initiators that enable CSMA/CA “listen before talk” (LBT) channel activity detection before sending each Data DPDU meet the requirements of ETSI EN 300 328 for adaptivity.

NOTE 4 These requirements are ETSI EN 300 328:2012 v1.8.1, 4.3.1.6.1 (FHSSM) and 4.3.2.5.2.2.1 (WBM) and related text. Although those requirements mention “adaptive modulation” the referenced modulation itself is fixed, not adaptive; instead it is use of the fixed modulation that is adaptive. Hence a more appropriate term is “adaptivity”.

Under ETSI EN 300 328,

- Tx-sequence- time is the transmitter-on time required to send a Data DPDU, which is ≤ 4,256 ms. In some cases it is also the transmitter-on time to send an ACK/NAK DPDU, which is ≤ 1 ms;
- Tx-gap-time is the minimum required interval of non-transmission between the end of one transmission and the beginning of the next transmission by the same device; and
- “dwell time” (DT) is the nominal time that a D-transaction initiator using FHSSM keeps its transmitter tuned to a given channel before changing to another channel.

NOTE 5 Tx-sequence-time and Tx-gap-time are defined in ETSI EN 300 328:2012 v1.8.1, 4.3.1.2 (FHSSM) and 4.3.2.3 (WBM). Dwell time, which applies to FHSSM, is defined to some extent in ETSI EN 300 328 v1.8.1, 3.1 under “frequency hopping spread spectrum” and in ETSI EN 300 328:2012 v1.8.1, 4.3.1.3.1. Dwell time is necessarily at least as large as Tx-sequence-time.

ETSI EN 300 328:2012 v1.8.1, 4.3.2.2.2 (WBM) imposes a power spectral density limit for WBM of 10 dBm/MHz EIRP. Due to the spectrum of the IEEE 802.15.4 2,4 GHz DSSS modulation, this constraint limits equipment operating under ETSI EN 300 328 v1.8.1’s WBM

regulations to a maximum transmit power (total for all active transmit chains after any antenna and beam-forming gain) of 20 mW (+13 dBm) EIRP.

ETSI EN 300 328:2012 v1.8.1, 4.3.1.1 (FHSSM) and 4.3.2.1 (WBM) limit maximum transmit power, after any antenna and beamforming gain, to 100 mW (+20 dBm) EIRP.

ETSI EN 300 328:2012 v1.8.1, 4.3.1.5 (FHSSM) and 4.3.2.4 (WBM) limit average transmit power of non-adaptive equipment, and of adaptive equipment operating in a non-adaptive mode, to 10 mW (+10 dBm) EIRP. Use of adaptivity removes this restriction on average transmit power.

When WBM without adaptivity is claimed, under ETSI EN 300 328:2012 v1.8.1, 4.3.2.3 each D-transaction-respondent in one timeslot is not permitted to initiate a D-transaction in the immediately-following timeslot unless the intervening period of non-transmission meets the minimum Tx-gap-time requirement of 3,5 ms, which is inherently greater than the Tx-sequence-time for any just-sent ACK/NAK DPDU.

Similarly, when FHSSM without adaptivity is claimed, under ETSI EN 300 328:2012 v1.8.1, 4.3.1.2 each D-transaction-respondent in one timeslot is not permitted to initiate a D-transaction in the immediately-following timeslot unless the intervening period of non-transmission meets the minimum Tx-gap-time requirement of 5 ms, which is inherently greater than the Tx-sequence-time for any just-sent DPDU.

When FHSSM with adaptivity is claimed, the ACK/NAK DPDUs that are sent by D-transaction respondents as immediate responses (within the same slot) to the Data DPDU sent by the D-transaction initiator can be considered “short control signaling” (SCS). While LBT is not required before transmitting SCS, under ETSI EN 300 328:2012 v1.8.1, 4.3.1.6.3.2 SCS is constrained to occupy no more than 10 % of the claimed dwell time. That restriction has an inverse impact on the minimum timeslot duration for the system, requiring the timeslot duration to be increased (and aggregate system throughput correspondingly decreased) relative to that otherwise required, just so that the channel occupancy of SCS (i.e., ACK/NAK DPDUs) in devices claiming conformance to FHSSM is never greater than 10 % of the claimed nominal dwell time.

The recommended alternative approach to meeting ETSI EN 300 328:2012 v1.8.1, 4.3.1.6.3.2 is to have each D-transaction-respondent dynamically mode-switch to operation in a non-adaptive mode while sending its ACK/NAK DPDU and for 5 ms thereafter (the mandated minimum Tx-gap-time), after which it reverts to the adaptivity mode of operation. It appears that the only significant consequence of such a temporary non-adaptive operating mode is that the responding device is not permitted to initiate a D-transaction in the immediately-following timeslot unless the intervening period of non-transmission meets the minimum Tx-gap-time requirement.

ETSI EN 300 328:2012 v1.8.1, 4.3.1.3.2 (FHSSM) requires that each cyclic channel-hopping sequence contain a minimum of 15 channels, whether idle or active. In terms of this standard, this requirement means that only dlmo.Ch entries (Table 160) whose size field has a value of 15 or greater are suitable for use in FHSSM mode under ETSI EN 300 328 v1.8.1. Therefore, when channel-hopping sequences with cycle lengths less than 15 are used, operation under ETSI EN 300 328 v1.8.1 necessarily conforms to the EN’s WBM regulations.

ETSI EN 300 328 v1.8.1 imposes no constraints on duty cycle for equipment whose maximum transmit power is ≤ 10 mW (+10 dBm) EIRP.

NOTE 6 In addition to its use during device provisioning, during which the intended configuration of end use can be set, this low maximum transmit power is adequate for short-distance use when site-specific considerations are unnecessary or inapplicable, such as within a laboratory or on and between tank cars in the consist of a moving train.

It appears that a device conforming to this standard can comply with the requirements of ETSI EN 300 328 v1.8.1 by being declared to operate in any one of six modes and configuring its dlmo.CountryCode (9.1.15.6) attributes, particularly bits 11 through 14, appropriately:

1) low-power WBM equipment, with dlmo.CountryCode.mode=0b"x0011x"; or
2) non-adaptive WBM equipment, with dlmo.CountryCode.mode=0b"x0001x"; or
3) adaptive WBM equipment, with dlmo.CountryCode.mode=0b"x0101x"; or
4) low-power FHSSM equipment, with dlmo.CountryCode.mode=0b"x1011x"; or
5) non-adaptive FHSSM equipment, with dlmo.CountryCode.mode=0b"x1001x"; or
6) adaptive FHSSM equipment that temporarily mode-switches to non-adaptive operation when operating as a D-transaction responder (i.e., to send an ACK/NAK DPDU) with dlmo.CountryCode.mode=0b"x1101x".

NOTE 7 Although adaptive FHSSM equipment that does not temporarily mode-switch is possible, the constraints induced on declared dwell time and thus minimum timeslot duration required to operate under that set of constraints make such a hypothetical operating category inferior to 6), due to the massively reduced system throughput that such overly-extended timeslots necessarily induce.

NOTE 8 If regulators determine that equipment conforming to this standard does not meet the full regulatory intent for one or more of the above six possible modes, operation under any of the remaining nodes is still possible.

Each of combinations 1) to 6) imposes a different set of constraints. Some are addressed automatically by all wireless devices that conform to this IEC standard, while others depend on the claimed operating mode. Whichever mode is selected and configured via the device's dlmo.CountryCode attribute, the device then operates in such a manner and takes whatever action is required to conform to those constraints.

Summarizing the above, the regulatory constraints that require self-monitoring are:

a) for operation in modes 1) and 4), limiting the maximum output power, $P_{out}Max$, to less than 10 mW (+10 dBm) EIRP;
b) for operation in modes 2) and 3), limiting the maximum output power, $P_{out}Max$, to 20 mW (+13 dBm) EIRP, which is 10 mW/MHz EIRP density for the signaling of IEEE 802.15.4 2,4 GHz DSSS;
c) for operation in modes 5 and 6, limiting the maximum output power, $P_{out}Max$, to 100 mW (+20 dBm) EIRP;
d) for operation in mode 2), limiting the total number of transmissions, both of Data DPDUs and of ACK/NAK DPDUs, such that the mean output power, $P_{out}Avg$, is 10 mW (+10 dBm) EIRP or less over every 1,0 s measurement interval;
e) for operation in mode 5), limiting the total number of transmissions, both of Data DPDUs and of ACK/NAK DPDUs, such that the mean output power, $P_{out}Avg$, is 10 mW (+10 dBm) EIRP or less over every measurement interval of 100 times the duration of one timeslot;
f) for operation in mode 6), limiting the total number of transmissions of ACK/NAK DPDUs such that the mean output power, $P_{out}Avg$, used while transmitting ACK/NAK DPDUs is 10 mW (+10 dBm) EIRP or less over every measurement interval of 100 times the duration of one timeslot;;

NOTE 9 The majority of channel occupancy by devices operating under mide 6) occurs when sending Data DPDUs. Such transmissions qualify as using adaptivity under 300 328 v1.8.1, 4.3.1.5, so constraint d) does not apply to them. However, devices operating under category 6) transmit ACK/NAK DPDUs in non-adaptive mode, to which constraint d) does apply per ETSI EN 300 328 v1.8.1, 4.3.1.5. Therefore it appears that only the totality of such ACK/NAK DPDUs transmitted by a device operating under mode 6) are subject to the constraint d) power limit. If that interpretation of ETSI EN 300 328 v1.8.1 is correct, then constraint d) will affect primarily backbone routers (BBRs), which in an automation WISN are largely receivers of process value-and-status publications and alert reports from WISN field devices. BBRs acting as transaction responders in duocast transaction may be impacted more than those BBRs not receiving duocasts.

g) for operation in mode 2) and 5), meeting the required Tx—gap-time of non-transmission after transmission of a Data DPDU;

NOTE 10 This constraint is met automatically whenever the slot duration is ≥ 8,508 ms in mode 2), or ≥ 9,256 ms in mode 5).

h) for operation in modes 2 and 6), meeting the required Tx-gap-time interval of non-transmission after transmission of an ACK/NAK DPDU.

For d), e), f), g) and h), the equipment dynamically monitors its recent activity to avoid transmitting whenever doing so would violate any of those constraints.

NOTE 11 While a system manager can schedule device activity pessimistically to ensure that d), e), f), g) and h) are always met, it is the device's own responsibility to monitor its recent activity and inhibit transmission when doing so would violate regulatory constraints. Thus the ultimate responsibility for operation of a collection of devices rests with the individual devices themselves, not some remote manager that could be subverted by a successful cyber attack on a single device.

Bibliography

IEC 61158 (all parts), *Industrial communication networks – Fieldbus specifications*

IEC 61499-4:2005, *Function blocks – Part 4: Rules for compliance profiles*

IEC 61512-1, *Batch control – Part 1: Models and terminology*

IEC 61804-3, *Function blocks (FB) for process control – Part 3: Electronic device description language (EDDL)*

IEC 62264-1:2013, *Enterprise-control system integration – Part 1: Models and terminology*

IEC/TS 62351-2:2008, *Power systems management and associated information exchange – Data and communications security – Part 2: Glossary of terms*

IEC/TR 62390:2005, *Common automation device – Profile guideline*

IEC/TS 62443-1-1:2009, *Industrial communication networks – Network and system security – Part 1-1: Terminology, concepts and models*

IEC 62591, *Industrial communication networks – Wireless communication network and communication profiles – WirelessHART™*

IEC 62601, *Industrial communication networks – Fieldbus specifications – WIA-PA communication network and communication profile*

IEC 62657-2, *Industrial communication networks – Wireless communication networks – Part 2: Coexistence management*

ISO/IEC 2375, *Information technology – Procedure for registration of escape sequences and coded character sets*

ISO/IEC 2382-14:1997, *Information technology – Vocabulary – Part 14: Reliability, maintainability and availability*

ISO/IEC 7498-1:1994 as corrected and reprinted in 1996, *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*

ISO/IEC 7498-2, *Information processing systems – Open systems interconnection – Basic reference model – Part 2: Security architecture*

ISO/IEC 7498-3:1997, *Information technology – Open Systems Interconnection – Basic Reference Model: Naming and addressing*

ISO/IEC 7498-4, *Information processing systems – Open systems interconnection – Basic reference model – Part 4: Management framework*

ISO/IEC 9646-7, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 7: Implementation Conformance Statements*

ISO/IEC 9796-2:2010, *Information technology – Security techniques – Digital signature schemes giving message recovery – Part 2: Integer factorization based mechanisms*

ISO/IEC 9797-1:2011, *Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher*

ISO/IEC 9797-2:2011, *Information technology – Security techniques – Message Authentication Codes (MACs) – Part 2: Mechanisms using a dedicated hash-function*

ISO/IEC 9798-1:2010, *Information technology – Security techniques – Entity authentication – Part 1: General*

ISO/IEC 10116:2006, *Information technology – Security techniques – Modes of operation for an n-bit block cipher*

ISO/IEC 10118-2, *Information technology – Security techniques – Hash-functions – Part 2: Hash-functions using an n-bit block cipher*

ISO/IEC 10118-3, *Information technology – Security techniques – Hash-functions – Part 3: Dedicated hash-functions*

ISO/IEC 10181-1:1996, *Information technology – Open Systems Interconnection – Security frameworks for open systems: Overview*

ISO/IEC 11770-1:2010, *Information technology – Security techniques – Key management – Part 1: Framework*

ISO/IEC 11770-2, *Information technology – Security techniques – Key management – Part 2: Mechanisms using symmetric techniques*

ISO/IEC 11770-3:2008, *Information technology – Security techniques – Key management – Part 3: Mechanisms using asymmetric techniques*

ISO/IEC 15408 (all parts), *Information technology – Security techniques – Evaluation criteria for IT security*

ISO/IEC 18028-3:2005, *Information technology – Security techniques – IT network security – Part 3: Securing communications between networks using security gateways*

ISO/IEC 18031:2011, *Information technology – Security techniques – Random bit generation*

ISO/IEC 18033-1:2005, *Information technology – Security techniques – Encryption algorithms – Part 1: General*

ISO/IEC 18033-2, *Information technology – Security techniques – Encryption algorithms – Part 2: Asymmetric ciphers*

ISO/IEC 19790:2012, *Information technology – Security techniques – Security requirements for cryptographic modules*

ISO/IEC 26907:2009, *Information technology – Telecommunications and information exchange between systems – High-rate ultra-wideband PHY and MAC standard*

ISO/IEC 27000:2014, *Information technology – Security techniques – Information security management systems – Overview and vocabulary*

ISO/IEC/IEEE 60559, *Information technology – Microprocessor Systems – Floating-Point arithmetic*

ISO 2382-12:1988, *Information processing systems – Vocabulary – Part 12: Peripheral equipment*

ISO 3166-1, *Codes for the representation of names of countries and their subdivisions – Part 1: Country codes*

ISO 11568-2:2012, *Financial services – Key management (retail) – Part 2: Symmetric ciphers, their key management and life cycle*

ISO 11568-4:2007, *Banking – Key management (retail) – Part 4: Asymmetric cryptosystems -- Key management and life cycle*

ISO 18435-2:2012, *Industrial automation systems and integration — Diagnostics, capability assessment and maintenance applications integration — Part 2: Descriptions and definitions of application domain matrix elements*

ISO 21188:2006, *Public key infrastructure for financial services – Practices and policy framework*

IEEE 802.1Q, *IEEE Standard for Local and metropolitan area networks – Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks*

IEEE 802.3, *IEEE Standard for Information technology-Specific requirements – Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications*

NOTE 1 ISO/IEC 8802-3 is based on IEEE 802.3, usually with some delay in publication.

IEEE 802.11:2012, *IEEE standards for information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 11: Wireless LAN medium access control (MAC) and physical layer (PhL) specifications*

NOTE 2 ISO/IEC 8802-11 is based on IEEE 802.11:2012, usually with some delay in publication.

IEEE 802.15.1:2005, *IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements. Part 15.1: Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs)*

NOTE 3 ISO/IEC 8802-15-1 is based on IEEE 802.15.1:2005, usually with some delay in publication.

IEEE Std 802.15.4e-2012, (Amendment to IEEE Std 802.15.4-2011), I*EEE Standard for Local and metropolitan area networks— Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer*

IEEE 802.16:2012, *IEEE Standard for local and metropolitan area networks–Part 16: Air interface for fixed broadband wireless access systems*

IERS conventions, *IERS technical note 32*

IETF RFC 1350, *The TFTP protocol, rev. 2*

IETF RFC 2347, *TFTP option extension*

IETF RFC 2348, *TFTP blocksize option*

IETF RFC 2349, *TFTP timeout interval and transfer size options*

IETF RFC 2525, *Known TCP implementation problems*

IETF RFC 3280, *Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile*

IETF RFC 5348, *TCP-friendly rate control (TFRC): Protocol specification*

IETF RFC 4949, *Internet security glossary, rev. 2*

ERC Recommendation 70-03, *Relating to the use of Short Range Devices (SRD), Annex 1 Band h and Annex 3 Band a*

ETSI EN 300 328 1 v1.8.1, *Electromagnetic compatibility and Radio spectrum Matters (ERM); Wideband transmission systems; Data transmission equipment operating in the 2,4 GHz ISM band and using wide band modulation techniques; Harmonized EN covering the essential requirements of article 3.2 of the R&TTE Directive*

ETSI EN 300 440 1 v1.6.1, *Electromagnetic compatibility and Radio spectrum Matters (ERM); Short range devices; Radio equipment to be used in the 1 GHz to 40 GHz frequency range; Part 1: Technical characteristics and test methods*

ETSI EN 300 440 2 v1.4.1, *Electromagnetic compatibility and Radio spectrum Matters (ERM); Short range devices; Radio equipment to be used in the 1 GHz to 40 GHz frequency range; Part 2: Harmonized EN covering essential requirements of article 3.2 of the R&TTE Directive*

ISC RSS 210, *Radio standards specification 210 – Low-power license-exempt radio communication devices (all frequency bands): Category I equipment*

ANSI X9.82-1, *Random number generation – Part 1: Overview and basic principles*

ISA Handbook of Measurement Equations and Tables, 2nd Edition,
ISBN 978-1-55617-946-4

ISA 100.11a, *Wireless Systems for Industrial Automation: Process Control and Related Applications*

ISA TR100.00.01-2006, *The Automation Engineer's Guide to Wireless Technology Part 1 – The Physics of Radio, a Tutorial*

[US] FIPS 186-3, *Digital Signature Standard (DSS)*

[US] FIPS 197, *Advanced encryption standard (AES)*

[US] FIPS 198, *The keyed-hash message authentication code (HMAC)*

[US] NIST SP 800-22, *A statistical test suite for random and pseudorandom number generators for cryptographic applications*

[US] NIST SP 800-38C, *Recommendation for block cipher modes of operation – The CCM mode for authentication confidentiality*

[US] NIST SP 800-56A, *Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography*

[US] NIST SP 800-57, *Recommendation for key management – Part 1: General*

[US] NIST SP 800-57, *Recommendation for key management – Part 2: Best practices for key management organization*

[US] NIST SP 800-88:2012, rev. 1, *Guidelines for media sanitization*

[US] Code of Federal Regulations (CFR) Title 47, Chapter I, *Part 15 – Telecommunication – Part 15: Radio frequency devices*

NAMUR Recommendation NE105, *Specifications for integrating fieldbus devices*

NAMUR Recommendation NE107, *Self-monitoring and diagnostics of field devices*

Guidelines for 64-bit Global Identifier (EUI-64™), available at http://standards.ieee.org/develop/regauth/tut/eui64.pdf.

HCF_SPEC-183, *Common Tables Specification*, available to members of the HART Communication Foundation, http://www.hartcomm.org

A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, *Handbook of applied cryptography*, ISBN 0-8493-8523-7, 1996

D. R. L. Brown, R. P. Gallant, S. A. Vanstone, *Provably secure implicit certificate schemes*, pp. 156-165 of ISBN 3-540-44079-8

F. Stajano, *The resurrecting duckling: What next?*, in *Proceedings of the 8th international workshop on security protocols*, B. Crispo, M. Roe, and B. Crispo, Eds., Lecture notes in computer science, Vol. 2133, Berlin: Springer-Verlag, April 2000.

F. Stajano, R. Anderson, *The resurrecting duckling: Security issues in ad-hoc wireless networks*, in Proceedings of the 7th international workshop on security protocols, B. Christianson, B. Crispo, J.A. Malcolm, and M. Roe, Eds., Lecture notes in computer science, Vol. 1796, Berlin: Springer-Verlag, 1999.

J. Jonsson, *On the security of CTR + CBC-MAC*, in *Proceedings of selected areas in cryptography – SAC 2002*, K. Nyberg, H. Heys, Eds. Lecture notes in computer science, Vol. 2595, pp. 76-93, Berlin: Springer, 2002.

J. Jonsson, *On the security of CTR + CBC-MAC, NIST mode of operation – Additional CCM documentation*, http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/ccm/ccm-ad1.pdf

PKIX, L. Bassham, R. Housley, W. Polk, *Algorithms and identifiers for the internet X.509 Public key infrastructure certificate and CRL profile*, ftp://ftp.isi.edu/in-notes/rfc3279.txt

P. Rogaway, D. Wagner, *A critique of CCM*, IACR ePrint Archive 2003-070, April 13, 2003

R. Housley, D. Whiting, N. Ferguson, *Counter with CBC-MAC (CCM)*, submitted to NIST., June 3, 2002

SOMMAIRE

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

RÉSEAUX INDUSTRIELS – RÉSEAU DE COMMUNICATION SANS FIL ET PROFILS DE COMMUNICATION – ISA 100.11A

AVANT-PROPOS

1) La Commission Electrotechnique Internationale (IEC) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de l'IEC). L'IEC a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. A cet effet, l'IEC – entre autres activités – publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de l'IEC"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec l'IEC, participent également aux travaux. L'IEC collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
2) Les décisions ou accords officiels de l'IEC concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de l'IEC intéressés sont représentés dans chaque comité d'études.
3) Les Publications de l'IEC se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de l'IEC. Tous les efforts raisonnables sont entrepris afin que l'IEC s'assure de l'exactitude du contenu technique de ses publications; l'IEC ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de l'IEC s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de l'IEC dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de l'IEC et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) L'IEC elle-même ne fournit aucune attestation de conformité. Des organismes de certification indépendants fournissent des services d'évaluation de conformité et, dans certains secteurs, accèdent aux marques de conformité de l'IEC. L'IEC n'est responsable d'aucun des services effectués par les organismes de certification indépendants.
6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
7) Aucune responsabilité ne doit être imputée à l'IEC, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de l'IEC, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de l'IEC ou de toute autre Publication de l'IEC, ou au crédit qui lui est accordé.
8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.

La Norme internationale IEC 62734 a été établie par le sous-comité 65C: Réseaux industriels, du comité d'études 65 de l'IEC: Mesure, commande et automation dans les processus industriels.

La présente norme internationale est basée sur l'ISA 100.11a:2011.

L'attention du lecteur est attirée sur le fait que l'Annexe V énumère tous les articles traitant des différences à caractère moins permanent inhérentes à certains pays, concernant le sujet de cette norme.

Cette première édition annule et remplace IEC/PAS 62734 parue en 2012. Cette édition constitue une révision technique.

Le texte de cette norme est issu des documents suivants:

FDIS	Rapport de vote
65C/778/FDIS	65C/788/RVD

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à l'approbation de cette norme.

Cette publication a été rédigée selon les Directives ISO/IEC, Partie 2.

Le comité a décidé que le contenu de cette publication ne sera pas modifié avant la date de stabilité indiquée sur le site web de l'IEC sous "http://webstore.iec.ch" dans les données relatives à la publication recherchée. A cette date, la publication sera

- reconduite,
- supprimée,
- remplacée par une édition révisée, ou
- amendée.

IMPORTANT – Le logo *"colour inside"* qui se trouve sur la page de couverture de cette publication indique qu'elle contient des couleurs qui sont considérées comme utiles à une bonne compréhension de son contenu. Les utilisateurs devraient, par conséquent, imprimer cette publication en utilisant une imprimante couleur.

0 Introduction

0.1 Généralités

La présente norme fournit des spécifications conformes au Modèle de référence de base OSI, ISO/IEC 7498–1, (par exemple: PhL, DL, etc.) et fournit également des spécifications de sécurité et de gestion (y compris la configuration de réseau et d'appareil) pour des appareils sans fil servant les classes d'utilisation 1 à 5 de l'Annexe C et potentiellement la classe 0, pour les appareils fixes, portatifs et mobiles.

La présente norme vise à assurer le fonctionnement fiable et sécurisé en mode sans fil pour les applications non critiques de contrôle, d'alerte, de commande de surveillance, de commande à boucle ouverte et de commande à boucle fermée. La présente norme définit une suite de protocoles, y compris la gestion de système, des considérations de passerelle et des spécifications de sécurité, pour une connectivité sans fil à bas débit avec des appareils fixes, portatifs et lentement mobiles, fonctionnant souvent sous des contraintes sévères d'énergie et de puissance. L'application est principalement axée sur les besoins de performance du contrôle et de la surveillance d'automation de processus où des latences de communication de bout en bout de l'ordre d'au moins 100 ms peuvent être tolérées.

Afin de satisfaire aux besoins des utilisateurs et opérateurs industriels du sans-fil, la technologie spécifiée dans le présent document fournit de la robustesse en la présence d'interférences rencontrées dans les environnements industriels sévères ou provoquées par des systèmes sans fil qui ne sont pas couverts par la présente norme. Conformément à la description donnée à l'Article 4, la présente norme traite de la coexistence avec d'autres appareils sans fil dans l'espace de travail industriel, tels que les téléphones cellulaires et les appareils basés sur l'IEC 62591 (basée sur WirelessHART™[1]), l'IEC 62601 (basée sur WIA-PA), l'IEEE 802.11:2012 (WiFi), l'IEEE 802.15, l'IEEE 802.16:2012 (WiMax) et d'autres normes applicables. De surcroît, la présente norme prend en charge l'interopérabilité des appareils conformes à la présente norme internationale, tels que décrits à l'Article 5, dans les aspects de fonctionnement couverts par la présente norme internationale.

La présente norme ne définit ni ne spécifie l'infrastructure d'une installation ou ses caractéristiques de sécurité ou de performances. Cependant, il est important que la sécurité de l'infrastructure de l'installation soit assurée par l'utilisateur final.

0.2 Structure du document

Le présent document est organisé en articles axés sur des fonctions de réseau et de couches de suite de protocoles uniques. Les articles décrivent le système, la gestion du système, la gestion de la sécurité, la couche physique, la couche liaison de données, la couche réseau, la couche transport, la couche d'application et la configuration. Des considérations génériques qui s'appliquent aux passerelles de protocoles sont également incluses, bien que les spécifications de passerelles de protocoles spécifiques ne le soient pas. Chaque article décrit une fonctionnalité ou une couche de protocoles et dicte le comportement indispensable pour un fonctionnement correct. Lorsqu'un article décrit des comportements relatifs à une autre fonction ou à une autre couche, une référence à l'autre article approprié est fournie pour informations complémentaires.

Les protocoles de communication obligatoires et facultatifs définis par le présent document se réfèrent à des protocoles natifs, alors que les protocoles utilisés par d'autres réseaux tels que les protocoles de communication de bus de terrain hérités se réfèrent à des protocoles étrangers.

---

[1] Propriété de HCF (HART Communication Foundation). Cette information est donnée à l'intention des utilisateurs de la norme et ne signifie nullement l'approbation ou la recommandation du propriétaire de la marque ou des produits associés. La conformité à ce profil n'exige pas l'utilisation de la marque déposée. L'utilisation des marques déposées exige l'obtention préalable d'autorisations auprès du propriétaire des marques.

0.3 Droits de propriété potentiellement applicables

La Commission Electrotechnique Internationale (IEC) attire l'attention sur le fait qu'il est déclaré que la conformité avec les dispositions du présent document peut impliquer l'utilisation de plusieurs brevets:

a) intéressant la cryptographie (asymétrique) sur courbes elliptiques traitée en 7.4.6 et en 7.2.2.3;

b) intéressant la synchronisation d'horloges et l'évaluation de la qualité de liaison, traitées en 9.1.9.3 et en 9.1.15;

c) intéressant des domaines de sujets non spécifiés;

d) intéressant la mise en service d'un réseau sans fil, ainsi que le choix et le routage de plusieurs passerelles.

L'IEC ne prend pas position quant à la preuve, à la validité et au domaine d'application de ces droits de propriété.

Les détenteurs de ces droits de propriété ont donné l'assurance à l'IEC qu'ils consentent à négocier des licences avec des demandeurs du monde entier, soit sans frais (gratuitement) soit à des termes et conditions raisonnables et non discriminatoires (RAND). A ce propos, les déclarations des détenteurs suivants de ces droits de propriété sont enregistrées à l'IEC.

Des informations peuvent être demandées à:

a)	Certicom Corporation 4701 Tahoe Blvd, Bldg A L4W 0B5 Mississauga, ON CANADA Attn: Patent licensing Dispositions de licence: vraisemblablement RAND Droits de propriété applicables: inconnus; non déclarés par le détenteur de droits de propriété	b)	NIVIS LLC 1000 Circle 75 Pkwy, Suite 300 Atlanta, GA 30339-6051 USA Attn: Patent licensing Dispositions de licence: RAND Droits de propriété applicables: – US 20100027437 – US 20100098204
c)	General Electric 1 Research Cir Schenectady, NY 12309-1027 USA Attn: Patent licensing Dispositions de licence: vraisemblablement RAND, réciprocité Droits de propriété applicables: inconnus; non déclarés par le détenteur de droits de propriété	d)	Yokogawa Electric Corporation 2-9-32 Nakachou, Musashina-shi Tokyo JAPAN Attn: Patent licensing Dispositions de licence: RAND, réciprocité Droits de propriété applicables: – JP 4129749 – US 8005514 – US 8031727 – US 8305927 – US 2009080394
Les détenteurs de droits de propriété, les droits de propriété et les dispositions de licence mentionnés ci-dessus correspondent à ceux déclarés dans la norme IEC tel qu'applicables à la norme IEC 62734, à compter de la date d'élaboration du présent texte.			

L'attention est attirée sur la possibilité que des éléments du présent document puissent être sujets à des droits de brevets autres que ceux identifiés ci-dessus. L'IEC ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de brevets et de ne pas avoir signalé leur existence.

L'ISO (http://www.iso.org/patents) et l'IEC (http://patents.iec.ch) tiennent à jour des bases de données en ligne sur les brevets relatifs à leurs normes. Les utilisateurs sont encouragés à consulter ces bases de données pour obtenir l'information la plus récente concernant les droits de propriété.

RÉSEAUX INDUSTRIELS – RÉSEAU DE COMMUNICATION SANS FIL ET PROFILS DE COMMUNICATION – ISA 100.11A

1 Domaine d'application

La présente norme internationale spécifie une méthode de fonctionnement fiable et sécurisé en mode sans fil pour les applications non critiques de contrôle, d'alerte, de commande de surveillance, de commande à boucle ouverte et de commande à boucle fermée. La présente norme définit une suite de protocoles, y compris la gestion de système, des considérations de passerelle et des spécifications de sécurité, pour une connectivité sans fil à bas débit avec des appareils fixes, portatifs et lentement mobiles, fonctionnant souvent sous des contraintes sévères d'énergie et de puissance. L'application dans la présente norme est principalement axée sur les besoins de performance du contrôle et de la surveillance d'automation de processus où des retards de communication de bout en bout de l'ordre de 100 ms peuvent être tolérés.

La présente norme spécifie ce qui suit:

- définition de service de la couche physique et spécification de protocole;
- définition de service de la couche liaison de données et spécification de protocole;
- définition de service de la couche réseau et spécification de protocole;
- définition de service de la couche transport et spécification de protocole;
- définition de service de la couche d'application et spécification de protocole, y compris la prise en charge pour la tunnellisation de protocoles et les passerelles;
- sécurité et gestion de la sécurité;
- mise en service et configuration;
- gestion de réseau; et
- profils additifs de rôles de communication (c'est-à-dire qu'un ou plusieurs peuvent être sélectionnés simultanément).

La fonctionnalité au-dessus de la couche d'application du Modèle de référence de base OSI, telle que ladite User Layer (couche d'utilisateur) et les différents profils pour la fonctionnalité en cette couche n'est pas adressée. Elle est toutefois brièvement débattue à l'Annexe A.

2 Références normatives

Les documents suivants sont cités en référence de manière normative, en intégralité ou en partie, dans le présent document et sont indispensables pour son application. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

NOTE Voir la Bibliographie pour les références non normatives.

ISO/IEC 646, *Information technology – ISO 7-bit coded character set for information interchange* (disponible en anglais seulement)

ISO/IEC 10731, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Modèle de référence de base – Conventions pour la définition des services OSI*

ISO/IEC 18033-3, *Information technology – Security techniques – Encryption algorithms – Part 3: Block ciphers* (disponible en anglais seulement)

ISO/IEC 19772, *Information technology – Security techniques – Authenticated encryption* (disponible en anglais seulement)

ANSI X9.63:2011, *Public Key Cryptography for the Financial Services Industry – Key Agreement and Key Transport Using Elliptic Curve Cryptography* (disponible en anglais seulement)

IETF RFC 2460:1998, *Internet Protocol, Version 6 (IPv6) Specification* (disponible en anglais seulement)

IETF RFC 2464, *Transmission of IPv6 Packets over Ethernet Networks* (disponible en anglais seulement)

IETF RFC 2529, *Transmission of IPv6 over IPv4 Domains without Explicit Tunnels* (disponible en anglais seulement)

IETF RFC 3168, *The Addition of Explicit Congestion Notification (ECN) to IP* (disponible en anglais seulement)

IETF RFC 4213, *Basic Transition Mechanisms for IPv6 Hosts and Routers* (disponible en anglais seulement)

IETF RFC 4291:2006, *IP Version 6 Addressing Architecture* (disponible en anglais seulement)

IETF RFC 4944, *Transmission of IPv6 Packets over IEEE 802.15.4 Networks* (disponible en anglais seulement)

IETF RFC 6282:2011, *Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks* (disponible en anglais seulement)

IETF RFC 6298, *Computing TCP's Retransmission Timer* (disponible en anglais seulement)

IEEE 802.15.4™:2011[2], *IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)* (disponible en anglais seulement)

SEC 1:2009, *Elliptic Curve Cryptography, version 2*, disponible à l'adresse http://www.secg.org (disponible en anglais seulement)

SEC 4, *Elliptic Curve Qu-Vanstone Implicit Certificate Scheme (ECQV), version 0.97*, disponible à l'adresse http://www.secg.org (disponible en anglais seulement)

3 Termes, définitions, abréviations, acronymes et conventions

Pour les besoins du présent document, les termes, définitions, abréviations et conventions suivants s'appliquent.

---

2 Propriété de l'IEEE, http://www.ieee.org.

3.1 Termes et définitions

3.1.1 Termes et définitions de couche (N) et autres issus du modèle de référence de base de l'interconnexion des systèmes ouverts

3.1.1.1
syntaxe abstraite
spécification d'unités de données de protocole (PDU) de couche (N) à l'aide de règles de notation indépendantes de la technique de codage utilisée pour les représenter

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 7.1.1.2, modifiée – généralisée à toutes les couches]

3.1.1.2
imputabilité
propriété qui garantit que les actions d'une entité ne peuvent être imputées qu'à cette entité

[SOURCE: ISO 7498-2:1989, 3.3.3]

3.1.1.3
acquittement
fonction de la couche (N) permettant à une entité (N) réceptrice d'informer l'entité (N) expéditrice de la réception d'une PDU (N)

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.8.1.16]

3.1.1.4
entité d'application
élément actif, à l'intérieur d'un processus d'application, comprenant un ensemble de capacités relatives à l'OSI et définies pour l'AL; cet ensemble correspond à un type d'entité d'application donné (sans autres capacités utilisées)

Note 1 à l'article: Il s'agit d'une légère spécialisation de l'entité (N), car la couche d'application (AL) inclut des fonctions applicatives non pertinentes pour l'OSI. Chaque entité d'application représente un et un seul processus dans l'environnement d'interconnexion des systèmes ouverts.

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 7.1.1.1]

3.1.1.5
gestion d'application
fonctions de l'AL ayant trait à la gestion des processus d'application OSI

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 8.1.1]

3.1.1.6
association
relation de coopération entre entités de système, habituellement dans le but de transférer de l'information entre elles

[SOURCE: IEC TS 62443-1-1:2009, 3.2.7]

3.1.1.7
association (N)
relation de coopération entre invocations d'entités (N)

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.3.1.1]

3.1.1.8
autorisation
attribution de droits, comprenant la permission d'accès sur la base de droits d'accès

[SOURCE: ISO 7498-2:1989, 3.3.10]

3.1.1.9
disponibilité
propriété d'être accessible et utilisable sur demande par une entité autorisée

[SOURCE: ISO 7498-2:1989, 3.3.11]

3.1.1.10
groupage
fonction accomplie par une entité (N) pour projeter plusieurs SDU (N) sur une PDU (N)

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.8.1.11]

3.1.1.11
connexion multipoint centralisée
connexion multipoint dans laquelle les données envoyées par l'entité associée à l'extrémité centrale de la connexion sont reçues par toutes les autres entités, alors que les données envoyées par une entité périphérique ne sont reçues que par l'entité centrale

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.8.1.2]

3.1.1.12
cryptogramme
données obtenues par l'utilisation du chiffrement, de sorte que le contenu sémantique des données résultantes n'est pas compréhensible

Note 1 à l'article: Voir texte en clair et texte clair.

Note 2 à l'article: Relativement à une PDU, le cryptogramme est de l'information dans une PDU qui est soumise à un obscurcissement par chiffrement, sous sa forme obscurcie après chiffrement avant déchiffrement.

[SOURCE: ISO 7498-2:1989, 3.3.1]

3.1.1.13
texte en clair
<génériques> données intelligibles dont la sémantique est compréhensible

[SOURCE: ISO 7498-2:1989, 3.3.15]

3.1.1.14
texte en clair
<spécifiques au protocole de communication> informations dans une PDU qui n'est pas soumise à un obscurcissement par chiffrement

Note 1 à l'article: Relativement à une PDU, le texte en clair est de l'information dans la PDU qui n'est pas soumise à un obscurcissement par chiffrement et qui, lorsqu'elle est présente dans la PDU, est toujours présente sous sa forme non obscurcie.

3.1.1.15
compromission
violation de la politique de sécurité d'un système informatique au cours de laquelle des programmes ou des données ont pu être modifiés, détruits, ou rendus accessibles à des entités non autorisées

[SOURCE: ISO/IEC 2382-8:1998, 08.05.11]

3.1.1.16
concaténation
fonction accomplie par une entité (N) pour projeter plusieurs PDU (N) sur une SDU (N-1)

Note 1 à l'article: Le groupage et la concaténation, bien que similaires (ils permettent tous deux de grouper des unités de données) servent à des fins différentes. Par exemple, la concaténation permet à la couche (N) de grouper une ou plusieurs unités PDU (N) d'acquittement avec une ou plusieurs unités PDU (N) contenant des données d'utilisateur, ce que la fonction de groupage ne peut pas réaliser. Noter également que ces deux fonctions peuvent être combinées permettant à la couche (N) d'effectuer des groupages et des concaténations.

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.8.1.13]

3.1.1.17
syntaxe concrète
aspects des règles utilisées dans la description formelle des données qui recouvrent une représentation spécifique de ces données

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 7.2.1.1]

3.1.1.18
confidentialité
propriété d'une information qui n'est ni disponible ni divulguée aux personnes, entités ou processus non autorisés

Note 1 à l'article: Dans un contexte général de sécurité de l'information, la confidentialité préserve les restrictions autorisées relatives à l'accès à des informations et à leur divulgation, y compris le moyen de préserver les informations privées individuelles et propriétaires.

[SOURCE: ISO 7498-2:1989, 3.3.16]

3.1.1.19
connexion (N)
association demandée par une entité de couche (N+1) pour le transfert de données entre deux entités (N+1) ou plus

Note 1 à l'article: L'association est établie par la couche (N) et fournit une identification explicite d'un ensemble de (N) transmissions de données et d'accord concernant les (N) services de transmission de données devant être fournis pour l'ensemble.

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.3.1.2]

3.1.1.20
extrémité de connexion (N)
terminaison d'une connexion (N) en un point d'accès aux services (N)

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.3.1.3]

3.1.1.21
identificateur d'extrémité de connexion (N)
identificateur de l'extrémité d'une connexion (N) destiné à identifier, en un SAP (N), la connexion (N) correspondante

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.4.1.5]

3.1.1.22
suffixe d'extrémité de connexion (N)
élément d'identificateur d'extrémité de connexion (N), unique dans le contexte d'un SAP (N)

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.4.1.6]

3.1.1.23
transmission en mode connexion (N)
transmission de données (N) dans le contexte d'une connexion (N)

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.3.1.17]

3.1.1.24
transmission en mode sans connexion (N)
transmission de données (N) hors du contexte d'une connexion (N) et qui n'est pas tenue de maintenir une quelconque relation logique entre les SDU (N)

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.3.1.18]

3.1.1.25
entités (N) correspondantes
entités (N) reliées par une connexion (N-1)

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.3.1.5]

3.1.1.26
analyse cryptographique
analyse d'un système cryptographique et/ou de ses entrées et sorties pour en déduire des variables confidentielles et/ou des données sensibles (y compris du texte en clair)

[SOURCE: ISO 7498-2:1989, 3.3.18]

3.1.1.27
intégrité des données
propriété assurant que des données n'ont pas été modifiées ou détruites de façon non autorisée

[SOURCE: ISO 7498-2:1989, 3.3.21]

3.1.1.28
authentification de l'origine des données
confirmation que la source des données reçues est telle que déclarée

[SOURCE: ISO 7489-2:1989, 3.3.22]

3.1.1.29
transmission de données (N)
fonctionnalité (N) transportant des SDU (N) d'une entité de couche (N+1) vers une ou plusieurs entités (N+1)

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.3.1.9]

3.1.1.30
dégroupage
fonction accomplie par une entité (N) pour identifier plusieurs SDU (N) contenues dans une PDU (N)

Note 1 à l'article: En l'absence d'erreur, le dégroupage est la fonction inverse de la fonction de groupage.

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.8.1.12]

3.1.1.31
connexion multipoint décentralisée
connexion multipoint dans laquelle les données envoyées par une entité associée à n'importe quelle extrémité de la connexion sont reçues par toutes les autres entités

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.8.1.3]

3.1.1.32
déchiffrement
opération inverse d'un chiffrement réversible

[SOURCE: ISO 7498-2:1989, 3.3.23]

3.1.1.33
démultiplexage
fonction accomplie par une entité (N) qui identifie les PDU (N) correspondant à plusieurs connexions (N) parmi les SDU (N-1) reçues sur une même connexion (N-1)

Note 1 à l'article: En l'absence d'erreur, le démultiplexage est la fonction inverse de la fonction de multiplexage.

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.8.1.5]

3.1.1.34
signature numérique
données ajoutées à une unité de données, ou transformation cryptographique d'une unité de données, permettant à un destinataire de prouver la source et l'intégrité de l'unité de données et protégeant contre la contrefaçon (par le destinataire, par exemple)

[SOURCE: ISO 7498-2:1989, 3.3.26]

3.1.1.35
entité (N)
élément actif dans sous-système (N), comportant un ensemble de capacités définies pour la couche (N) et correspondant à un type donné d'entité (N) (sans que soient utilisées d'autres capacités)

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.2.1.11]

3.1.1.36
invocation d'entité (N)
utilisation particulière d'une partie ou de l'ensemble des capacités d'une entité (N) donnée (sans que soient utilisées d'autres capacités)

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.2.1.12]

3.1.1.37
type d'entité (N)
description d'une classe d'entités (N) en termes d'ensemble de capacités définies pour la couche (N)

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.2.1.10]

3.1.1.38
information de contrôle d'interface (N)
information transférée en local entre une entité (N+1) et une entité (N) pour coordonner leur travail commun

3.1.1.39
couche (N)
subdivision de l'architecture OSI, constituée des sous-systèmes de rang (N)

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.2.1.2]

3.1.1.40
gestion de couche (N)
fonctions relatives à la gestion de la couche (N), effectuées en partie dans la couche (N) elle-même conformément au protocole (N) de la couche (activités telles que le contrôle d'erreurs et l'activation), et en partie par un sous-ensemble de la gestion-système

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 8.1.6]

3.1.1.41
connexion multipoint
connexion comportant plus de deux extrémités de connexion

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.3.1.4]

3.1.1.42
multiplexage
fonction accomplie par une entité (N) permettant à une seule connexion (N-1) de prendre en charge plusieurs connexions (N)

Note 1 à l'article: Le terme multiplexage est également utilisé dans un sens plus restrictif pour désigner la fonction accomplie par l'entité (N) expéditrice, alors que le terme démultiplexage sert à désigner la fonction accomplie par l'entité (N) destinataire.

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.8.1.4]

3.1.1.43
mot de passe
information d'authentification confidentielle, habituellement composée d'une chaîne de caractères

[SOURCE: ISO 7498-2:1989, 3.3.39]

3.1.1.44
entités (N) homologues
entités appartenant à la même couche (N)

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.2.1.3]

3.1.1.45
authentification de l'entité homologue
confirmation qu'une entité homologue d'une association est bien l'entité déclarée

[SOURCE: ISO 7489-2:1989, 3.3.40]

3.1.1.46
protocole (N)
ensemble de règles et de formats (sémantiques et syntaxiques) déterminant le comportement de communication des entités (N) lorsqu'elles exécutent les fonctions (N)

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.2.1.9]

3.1.1.47
informations d'adressage du protocole (N)
éléments des informations de contrôle (N)-PCI qui contiennent les informations relatives à l'adressage

[SOURCE: ISO/IEC 7498-3:1997, 3.4.20]

3.1.1.48
information de contrôle protocolaire (N)
information échangée entre entités (N), pour coordonner leur travail commun

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.6.1.1]

3.1.1.49
unité de données de protocole (N)
unité de données spécifiée dans un protocole (N) et comportant des informations de contrôle protocolaires (N) et, éventuellement, des données d'utilisateur (N)

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.6.1.3]

3.1.1.50
identificateur de version de protocole (N)
identificateur véhiculé entre entités (N) correspondantes permettant de sélectionner la version d'un protocole (N)

Note 1 à l'article: La définition d'un nouvel identificateur de version de protocole (N) suppose la connaissance minimale du protocole (N) identifié par l'identificateur de version de protocole (N) précédent. Dans les cas où une telle connaissance minimale ne peut pas être réalisée, on considère que les protocoles (N) sont indépendants et différents.

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.8.1.18]

3.1.1.51
qualité de service
<générique> effet collectif de la performance d'un service qui détermine le degré de satisfaction d'un utilisateur du service

Note 1 à l'article: La qualité de service est caractérisée par l'effet conjugué de la qualité d'implémentation du service, de sa facilité d'utilisation, de sa performance de fonctionnement, de l'intégrité du service et d'autres facteurs propres à chaque service.

Note 2 à l'article: L'ISO définit la "qualité" comme l'aptitude d'un produit ou d'un service à satisfaire les besoins des utilisateurs.

[SOURCE: IEC 61907:2009, 3.1.15]

3.1.1.52
qualité de service
<service de liaison de données> paramètres négociés pour une liaison, y compris

- la priorité;
- les fenêtres temporelles pour la messagerie de contrôle;
- acceptabilité de la livraison de message en dérangement; et
- acceptabilité de la livraison de message par incréments partiels

3.1.1.53
réassemblage
fonction accomplie par une entité (N) pour projeter plusieurs unités de données de protocole PDU (N) sur une unité de données de service SDU (N)

Note 1 à l'article: En l'absence d'erreur, le réassemblage est la fonction inverse de la fonction de segmentation.

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.8.1.10]

3.1.1.54
recombinaison
fonction accomplie par une entité (N) identifiant les unités de données de protocole (N) correspondant à une même connexion (N) parmi les unités de données de service (N-1) reçues sur plusieurs connexions (N-1)

Note 1 à l'article: En l'absence d'erreur, la recombinaison est la fonction inverse de la fonction d'éclatement.

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.8.1.7]

3.1.1.55
relais (N)
fonction (N) au moyen de laquelle une entité (N) retransmet à une autre entité homologue (N) les données reçues d'une entité homologue (N)

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.3.1.6]

3.1.1.56
réinitialisation
fonction mettant les entités (N) correspondantes à un état prédéfini, avec possibilité de perte ou de duplication de données

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.8.1.17]

3.1.1.57
étiquette de sécurité
marque liée à une ressource dénommant ou désignant les attributs de sécurité de cette ressource (cette ressource peut être une unité de données)

[SOURCE: ISO 7498-2:1989, 3.3.49]

3.1.1.58
segmentation
fonction accomplie par une entité (N) pour projeter une SDU (N) sur plusieurs PDU (N).

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.8.1.9]

3.1.1.59
sélecteur (N)
partie d'une adresse (N) qui est spécifique au sous-système (N) adressé, c'est-à-dire qui identifie un ou plusieurs SAP (N) dans un système ouvert d'extrémité, une fois que ce système ouvert d'extrémité a été identifié de manière non ambiguë

Note 1 à l'article: Comme le système ouvert d'extrémité est implicitement connu au niveau de la couche Réseau, les sélecteurs (N) sont utilisés au-dessus de la couche Réseau (ainsi que les informations locales) de manière à adresser l'entité (N+1) souhaitée au sein du système ouvert. Les valeurs des sélecteurs (N) sont échangées entre les systèmes ouverts dans le cadre du PAI (N).

[SOURCE: ISO/IEC 7498-3:1997, 6.2.3]

3.1.1.60
séparation
fonction accomplie par une entité (N) pour identifier plusieurs PDU (N) contenues dans une SDU (N-1)

Note 1 à l'article: En l'absence d'erreur, la séparation est la fonction inverse de la fonction de concaténation.

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.8.1.14]

3.1.1.61
séquencement
fonction assurée par la couche (N) pour conserver l'ordre des SDU (N) remises à la couche (N)

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.8.1.15]

3.1.1.62
service (N)
capacité fondamentale de la couche (N) et des couches inférieures à celle-ci, offerte aux entités (N+1) à la frontière entre la couche (N) et la couche (N+1)

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.2.1.5]

3.1.1.63
point d'accès au service (N)
point où les services (N) sont fournis par une entité (N) à une entité (N+1)

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.2.1.8]

3.1.1.64
adresse de point d'accès au service (N)
adresse (N) utilisée pour identifier un seul SAP (N)

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.4.1.2]

3.1.1.65
unité de données de service (N)
informations dont l'identité est préservée pendant leur transfert entre entités (N+1) homologues, et qui ne sont pas interprétées par les entités (N)

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.6.1.4]

3.1.1.66
éclatement
fonction de la couche (N) permettant d'utiliser plusieurs connexions (N-1) pour prendre en charge une même connexion (N)

Note 1 à l'article: Le terme éclatement est également utilisé dans un sens plus restrictif pour voir la fonction accomplie par l'entité (N) expéditrice, le terme recombinaison servant à voir la fonction accomplie par l'entité (N) destinataire

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.8.1.6]

3.1.1.67
gestion de systèmes
fonctions de l'AL relatives à la gestion de diverses ressources OSI et de leurs états dans toutes les couches de l'architecture OSI

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 8.1.4]

3.1.1.68
syntaxe de transfert
syntaxe abstraite et concrète utilisée dans le transfert des données entre systèmes ouverts

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 7.2.1.2]

3.1.1.69
processus d'application utilisateur
processus actif à la partie la plus élevée de l'AL qui est l'utilisateur des services OSI

Note 1 à l'article: Les aspects d'un processus d'application utilisateur (UAP) qui sont à prendre en compte pour l'OSI sont représentés par une ou plusieurs entités d'application, d'un ou plusieurs types d'entités d'application, conformément à l'ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 7.1.2.2 et 7.1.2.3.

Note 2 à l'article: L'ensemble des UAP est parfois appelé "couche d'utilisateur", même l'ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 7.1.2.1 énonce que l'AL n'a pas de frontière commune avec une couche de niveau supérieur. Dans le Modèle de référence de base OSI, l'AL inclut les UAP.

3.1.1.70
données d'utilisateur (N)
données transférées entre entités (N) pour le compte d'entités (N+1) qu'elles desservent

[SOURCE: ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, 5.6.1.2]

3.1.2 Autres termes et définitions

NOTE Les sources des définitions qui ne sont pas autrement référencées par la présente norme peuvent être consultées dans la Bibliographie.

3.1.2.1
contrôle d'accès
moyens pour assurer que l'accès aux actifs est autorisé et limité selon les exigences propres à la sécurité et à l'activité métier

[SOURCE: ISO/IEC 27000:2014, 2.1]

3.1.2.2
adaptateur
spécialisation d'un appareil qui convertit en interne le protocole de l'appareil ou des appareils hérités connectés dans celui d'un appareil réseau conforme à la présente norme

3.1.2.3
alarme
condition qui conserve l'état jusqu'à ce que la condition soit éliminée, rapportée à la suite d'un changement d'état

EXEMPLE L'occurrence d'une alarme ou d'une condition de retour à la normale qui est d'importance considérable pour un UAP correspondant.

3.1.2.4
alerte
action de rapporter une condition d'événement ou une condition d'alarme

3.1.2.5
application finale
application
système ou problème auquel un ordinateur est appliqué

3.1.2.6
programme d'application
application
programme qui fournit une fonctionnalité à des utilisateurs finals

3.1.2.7
couche d'application
application
couche de protocoles la plus élevée dans le modèle de référence de base ISO/IEC où les types d'applications sont classifiés selon la criticité

3.1.2.8
processus d'application
élément qui accomplit le traitement de l'information pour une application particulière

3.1.2.9
algorithme à clés asymétriques
algorithme de cryptographie à clés asymétriques
algorithme de cryptographie à clés publiques
<sécurité de l'information> algorithme pour réaliser le chiffrement ou le déchiffrement correspondant dans lequel les clés utilisées pour le chiffrement et le déchiffrement sont différentes

[SOURCE: ISO/IEC 10181-1:1996, 3.3.1]

3.1.2.10
authentification
<sécurité de l'information> vérification de l'identité d'un utilisateur, d'un processus ou d'un appareil, souvent comme condition préalable pour accorder l'accès à des ressources dans un système d'information

Note 1 à l'article: Voir authentification de l'origine des données (3.1.1.28) et/ou authentification de l'entité homologue (3.1.1.45)

3.1.2.11
code d'authentification
<sécurité de l'information> somme de contrôle, complète ou tronquée, basée sur une fonction de sécurité appropriée

Note 1 à l'article: Voir 3.1.2.97.

Note 2 à l'article: Il est également appelé code d'authentification de message (MAC). Il est appelé code d'intégrité de message (MIC) lorsqu'il est utilisé dans des contextes où l'acronyme MAC possède une autre définition, telle que dans les normes relatives aux réseaux locaux.

[SOURCE: ISO/IEC 19790:2012, 3.8, modifiée par le préfixe "complète ou tronquée"]

3.1.2.12
réseau dorsal
sous-réseau dorsal
réseau non spécifié par la présente norme, utilisant généralement la technologie réseau IPv6 or IPv4, qui est utilisé pour le routage entre le réseau de capteurs industriels (et d'actionneurs) sans fil (WISN) de la présente norme et

a) des appareils dorsaux qui sont spécifiés en partie par la présente norme, tels que les gestionnaires de système, les gestionnaires de sécurité et les passerelles de protocole;
b) des appareils qui prennent nativement en charge les protocoles TL, AL et de gestion sans fil de la présente norme; et
c) d'autres routeurs dorsaux résidant sur le même sous-réseau dorsal.

3.1.2.13
routeur dorsal
routeur qui retransmet entre le réseau sans fil de la présente norme et un réseau dorsal de haut débit

3.1.2.14
secours
procédure, technique ou matériel utilisés pour aider à recouvrer des données perdues ou détruites, ou pour maintenir un système en fonctionnement

[SOURCE: ISO 2382-12:1988, 12.01.17]

3.1.2.15
largeur de bande
<domaine analogique> différence numérique entre les fréquences supérieure et inférieure d'une bande de fréquences

Note 1 à l'article: La bande passante analogique est exprimée en Hz.

3.1.2.16
largeur de bande
<domaine numérique> quantité de données qui peut être passée le long d'une voie de communications pendant une durée donnée

Note 1 à l'article: La bande passante numérique est exprimée en bit/s.

3.1.2.17
canal noir
voie de communication d'un système de sécurité qui ne fournit aucune fonctionnalité de sûreté en plus de sa capacité élémentaire de communication

3.1.2.18
liste noire
liste des voies RF sur lesquelles l'émission est interdite

Note 1 à l'article: Une liste noire est temporaire ou permanente, locale ou à l'échelle du réseau.

3.1.2.19
cryptage par blocs
<sécurité de l'information> primitive cryptographique qui utilise une clé symétrique pour créer une permutation pseudo-aléatoire, dépendant d'une clé, d'une chaîne de bits de taille fixe

3.1.2.20
diffusion
émission destinée à tous les nœuds

Note 1 à l'article: La réception d'une diffusion est souvent limitée à des couches spécifiques, par exemple la couche MAC ou la couche réseau.

Note 2 à l'article: De nombreux protocoles de couche ne fournissent pas d'acquittement pour les diffusions.

3.1.2.21
syntaxe de transfert canonique
syntaxe de transfert complète
codage complet d'un objet en vue du transfert entre des appareils, avant toute compression

3.1.2.22
chiffre
<sécurité de l'information> technique cryptographique utilisée pour protéger la confidentialité des données, constituée de trois processus constitutifs: un algorithme de chiffrement, un algorithme de déchiffrement et une méthode de création de clés

[SOURCE: attribuée dans d'autres normes ISO/IEC à une édition inconnue de l'ISO/IEC 18033-1, modifiée – légèrement modifiée pour la lisibilité.]

3.1.2.23
coexistence
aptitude de plusieurs systèmes à accomplir leurs tâches dans un environnement ouvert lorsqu'ils peuvent ou peuvent ne pas utiliser un jeu similaire de règles

3.1.2.24
syntaxe de transfert compressée
codage d'un objet en vue du transfert entre des appareils, après toute compression

3.1.2.25
latence des communications
délai entre un événement de communication initial, tel que réaliser une demande de transmission, et un événement de communication ultérieur associé, tel que la livraison d'un message à un destinataire prévu

3.1.2.26
fiabilité des communications
degré auquel les messages atteignent leur destination finale à l'état complet et non corrompu et dans un délai de latence acceptable

3.1.2.27
débit des communications
capacité d'un canal de communication, généralement mesurée en messages/s ou octets/s

3.1.2.28
option de construction
ensemble de caractéristiques qu'un concepteur d'appareil peut choisir d'inclure dans un appareil ou d'exclure d'un appareil

3.1.2.29
contrat
accord entre le gestionnaire de système et un appareil dans le réseau impliquant l'allocation de ressources du réseau par le gestionnaire de système pour prendre en charge un besoin particulier de communication de l'appareil en question

3.1.2.30
algorithme cryptographique
<sécurité de l'information> algorithme basé sur la science de la cryptographie, y compris les algorithmes de chiffrement, les algorithmes de hachage cryptographique, les algorithmes de signature numérique et les algorithmes d'agrément/concordance de clés

Note 1 à l'article: Des exemples d'algorithmes cryptographiques sont les chiffrements par blocs, les chiffrements continus et les hachages à clés. Un hachage sans clé n'est pas formellement un algorithme cryptographique, bien qu'il soit souvent une fonction unidirectionnelle qui a une résistance aux attaques et qu'il soit souvent construit à partir d'un algorithme cryptographique avec une clé fixe. La famille de hachages SHA est ainsi construite.

[SOURCE: IEC TS 62443-1-1:2009, 3.2.34, modifiée – une note a été ajoutée]

3.1.2.31
clé cryptographique
clé
<sécurité de l'information> valeur mathématique qui est utilisée

a) dans un algorithme pour produire un cryptogramme à partir d'un texte en clair ou vice versa, et

b) pour déterminer le fonctionnement d'une fonction cryptographique (par exemple: la production synchronisée d'un support de codage), ou d'un calcul ou d'une validation d'une signature numérique

[SOURCE: IEC TS 62351-2:2008, 2.2.64]

3.1.2.32
composant clé cryptographique
composant clé
<sécurité de l'information> paramètre(s) utilisé(s) dans une fonction de sécurité pour accomplir une fonction cryptographique

[SOURCE: ISO/IEC 19790:2012, 3.24]

3.1.2.33
module cryptographique
<sécurité de l'information> ensemble de matériel, logiciel et/ou firmware qui met en œuvre des fonctions de sécurité appropriées et est contenu dans la limite cryptographique

[SOURCE: ISO/IEC 19790:2012, 3.25]

3.1.2.34
période cryptographique
<sécurité de l'information> intervalle de temps pendant lequel une clé spécifique peut être utilisée ou dans laque les clés pour un système ou une application donné(e) peuvent rester en vigueur

[SOURCE: ISO/IEC 19790:2012, 3.25]

3.1.2.35
authenticité des données
<sécurité de l'information> assurance relative à la source d'informations

[SOURCE: IEEE 802-15-4:2011, 3.1]

3.1.2.36
clé de données
clé d'authentification de données
clé de chiffrement de données
<sécurité de l'information> clé cryptographique utilisée pour le chiffrement, le déchiffrement ou l'authentification des données

[SOURCE: ISO/IEC 11568-2:2012, 3.5]

3.1.2.37
option de déploiement
ensemble de caractéristiques qu'un concepteur d'appareil inclut dans un appareil, mais que l'utilisateur final ou son agent (un gestionnaire de sécurité réseau, par exemple) peut choisir d'employer ou de ne pas employer

3.1.2.38
clé dérivée
<sécurité de l'information> clé symétrique qui est dérivée d'une clé symétrique antérieure

Note 1 à l'article: De telles clés sont utilisables pour limiter la période cryptographique de toute clé individuelle tout en satisfaisant aux exigences d'archivage de clé, à condition que la clé indépendante à partir de laquelle la clé dérivée a été dérivée (éventuellement par plusieurs productions de dérivation) ait préalablement satisfait à ces exigences d'archivage.

3.1.2.39
destruction de clé
destruction
<sécurité de l'information> abrogation ou destruction physique d'un support de codage afin qu'il ne puisse pas être récupéré

3.1.2.40
générateur aléatoire de bits déterministe
<sécurité de l'information> processus utilisé pour produire une série imprévisible de bits qui sont aléatoires dans le sens qu'il n'existe pas de façon de décrire la sortie du générateur qui soit plus efficace que d'énumérer simplement chaque chaîne entière de sortie

Note 1 à l'article: Les générateurs aléatoires de bits déterministes ont des propriétés prouvables. L'imprévisibilité de leur sortie dépend de l'imprévisibilité de leur germe initial et, pour les générateurs hybrides, de la vitesse à laquelle la nouvelle entrée imprévisible (d'entropie élevée) est incluse par rapport au nombre de bits de sortie générés. Voir la note à l'article 1 en 3.1.2.102, générateur aléatoire de bits non déterministe, pour les sources communes d'une telle imprévisibilité.

3.1.2.41
objet de gestion de sécurité d'appareil
logiciel d'application au sein d'un appareil qui agit comme homologue local d'un gestionnaire de sécurité

3.1.2.42
duodiffusion
variante de monodiffusion, dans laquelle un second destinataire est programmé pour entendre par hasard la DPDU et fournir un second acquittement dans une seule transaction D

Note 1 à l'article: La duodiffusion est montrée graphiquement à la Figure 84.

3.1.2.43
clé chiffrée
<sécurité de l'information> clé cryptographique qui a été chiffrée à l'aide d'une fonction de sécurité appropriée avec une clé de chiffrement de clé

Note 1 à l'article: Ce processus est utilisé pour dissimuler la valeur de la clé sous-jacente en texte en clair.

[SOURCE: ISO/IEC 19790:2012, 3.36]

3.1.2.44
chiffrement
<sécurité de l'information> opération réversible réalisée par un algorithme cryptographique convertissant des données en cryptogramme de façon à cacher le contenu informationnel des données

[SOURCE: ISO/IEC 9798-1:2010, 3.13, modifiée – traduite en français]

3.1.2.45
entité
individu (personne), organisation, appareil ou processus

3.1.2.46
clé éphémère
<sécurité de l'information> clé cryptographique qui est générée pour chaque exécution d'un processus d'établissement de clé et qui satisfait à d'autres exigences relatives au type de clé (par exemple: unique à chaque échange de message ou session)

Note 1 à l'article: Dans certains cas, les clés éphémères sont utilisées plus d'une fois, dans une même session (applications en multidiffusion, par exemple) où l'émetteur génère une seule clé éphémère par message et la clé privée (de cette paire) est combinée séparément avec chaque clé publique de destinataire.

3.1.2.47
événement
condition transitoire (c'est-à-dire, sans état), utilisée pour rapporter lorsque quelque chose s'est produit

EXEMPLE L'occurrence d'une alarme ou d'une condition de retour à la normale qui est d'importance considérable pour un UAP correspondant.

3.1.2.48
appareil de terrain
appareil physique conçu pour répondre aux rigueurs de l'exploitation d'installation qui communique par des DPDU et des protocoles de couche supérieure conformes à la présente norme

Note 1 à l'article: Y sont inclus les appareils de routage, les capteurs et les actionneurs.

3.1.2.49
réseau de terrain
configuration d'au moins deux appareils de terrain reliés entre eux par le protocole sans fil défini par la présente norme

3.1.2.50
routeur de terrain
routeur qui est également un appareil de terrain (un routeur dorsal, par exemple) existant au sein du réseau de terrain

3.1.2.51
communication d'application de protocole étranger
acheminement optimisé des PDU ou de portions des PDU à partir d'un premier protocole au sein d'un second protocole par l'usage sélectif de techniques de mise en cache, de compression, de conversion d'adresses et techniques de proxy

3.1.2.52
fragmenter, verbe
segmenter
fractionner ou séparer en parties disjointes contiguës

Note 1 à l'article: Fragment et fragmentation sont les termes utilisés par l'IETF dans les spécifications de protocole internet pour décrire les concepts OSI de segment et segmentation. Dans la présente norme, les termes "fragment" et "segment" sont essentiellement synonymes, le terme "fragment" étant généralement utilisé au niveau de la couche réseau et parfois sur d'autres couches de protocoles, tandis que le terme "segment" est généralement utilisé au niveau de la couche application et parfois sur d'autres couches de protocoles.

3.1.2.53
fragment, nom
segment
l'une des unités de données de protocole (N) résultant de l'opération de segmentation

Note 1 à l'article: Fragment et fragmentation sont les termes utilisés par l'IETF dans les spécifications de protocole internet pour décrire les concepts OSI de segment et segmentation. Dans la présente norme, les termes "fragment" et "segment" sont essentiellement synonymes, le terme "fragment" étant généralement utilisé au niveau de la couche réseau et parfois sur d'autres couches de protocoles, tandis que le terme "segment" est généralement utilisé au niveau de la couche application et parfois sur d'autres couches de protocoles.

3.1.2.54 passerelle
rôle (d'un appareil) qui agit comme un convertisseur de protocole entre une AE conforme à la présente norme et d'autres AE différentes

3.1.2.55
code d'authentification de message fondée sur un hachage
<sécurité de l'information> code d'authentification de message qui utilise une fonction de hachage à clé appropriée

Note 1 à l'article: Cette définition est généralisée à partir de la seconde construction de l'ISO/IEC 9797-2, qui se réfère à a construction HMAC également spécifiée dans [US] FIPS 198.

[SOURCE: ISO/IEC 9797-2:2011, modifiée – généralisée.]

3.1.2.56
fonction de hachage
<sécurité de l'information> fonction qui projette des chaînes de bits sur des chaînes de bits de taille fixe, en satisfaisant aux deux propriétés suivantes: pour une sortie donnée, il est irréalisable par le calcul de trouver une entrée qui correspond à cette sortie; pour une entrée donnée, il est irréalisable par le calcul de trouver une seconde entrée qui correspond à la même sortie

Note 1 à l'article: La faisabilité par le calcul dépend des exigences de sécurité et de l'environnement spécifiques.

Note 2 à l'article: Voir la note à l'article 1 en 3.1.2.30.

[SOURCE: ISO/IEC 9796-2:2010, 3.6, modifiée – des notes à l'article ont été ajoutées.]

3.1.2.57
valeur de hachage
<sécurité de l'information> résultat, complet ou tronqué, obtenu par l'application d'une fonction de hachage à des informations

3.1.2.58
identité
caractère ou personnalité distinctive d'un individu ou d'une entité

3.1.2.59
clé indépendante
<sécurité de l'information> clé symétrique qui est dérivée d'une source de bits d'entropie élevée et non d'une clé antérieure

3.1.2.60
infrastructure
structures techniques prenant en charge des communications de données au sein d'une installation

EXEMPLE Parties du réseau IT d'une installation, utilisant peut-être IEEE 802.3 ou IEEE 802.11, ou IEC 61158 Type 10 (PROFInet) ou IEC 61158 Type 9 (Foundation™ Fieldbus HSE).[3]

3.1.2.61
vecteur d'initialisation
<sécurité de l'information> bloc de bits qui est exigé pour permettre à un chiffre cryptographique dans un mode continu de fonctionnement de produire un flux unique indépendant vis-à-vis des autres flux produits sous la même clé de chiffrement

3.1.2.62
interopérables
capables de fonctionner ensemble pour accomplir un rôle spécifique dans un ou plusieurs programmes d'application distribués

Note 1 à l'article: Dans ce cas, les paramètres et leurs fonctionnalités applicatives s'ajustent à la fois de manière syntaxique et sémantique. L'interopérabilité est accomplie lorsque les appareils prennent en charge des ensembles complémentaires de paramètres et fonctions appartenant au même profil.

[SOURCE: IEC TR 62390:2005, 6.2.2.6, modifiée – adaptée.]

---

3 PROFInet et Foundation Fieldbus sont les marques commerciales de diverses organisations commerciales. Cette information est donnée à l'intention des utilisateurs de la présente norme et ne signifie nullement que l'IEC approuve ou recommande l´emploi exclusif du (des) produit(s) ainsi désigné(s). La conformité à ce profil n'exige pas l'utilisation de la marque déposée. L'utilisation des marques déposées exige l'obtention préalable d'autorisations auprès du propriétaire des marques.

3.1.2.63
interfonctionnables
capables de transférer des paramètres entre des correspondants

Note 1 à l'article: Outre le protocole de communication, l'interface de communication et l'accès aux données, les types de données de paramètres sont les mêmes.

[SOURCE: IEC TR 62390:2005, 6.2.2.5, modifiée – adaptée.]

3.1.2.64
protocole Kerberos
protocole d'authentification réseau spécifique qui permet à des individus communiquant sur un réseau non sécurisé de se prouver l'un à l'autre leur identité d'une manière sécurisée

3.1.2.65
agrément de clé
<sécurité de l'information> processus d'établissement d'une clé secrète partagée entre des entités de telle manière qu'aucune de celles-ci ne puisse déterminer la valeur de la clé en question

[SOURCE: ISO/IEC 11770-1:2010, 2.13]

3.1.2.66
archivage de clé
archive de gestion de clés
<sécurité de l'information> système de chiffrement avec une fonctionnalité de déchiffrement complémentaire qui permet à des personnes autorisées dans certaines conditions prescrites, de déchiffrer un cryptogramme avec l'aide d'informations fournies par une ou plusieurs parties de confiance qui détiennent des clés spéciales de récupération de données

3.1.2.67
centre de clés
<sécurité de l'information> processus de distribution centralisée de clés, habituellement un système informatique distinct, qui utilise des clés de chiffrement de clés (clés principales) pour chiffrer et distribuer des clés T dont a besoin une communauté d'utilisateurs

Note 1 à l'article: Les centres de clés sont généralement certifiés, traçables à une agence d'essais indépendante agréée, comme satisfaisant aux exigences de l'ISO/IEC 19790 (similaire à FIPS 140-2) pour un module cryptographique de niveau 3 ou de niveau 4.

3.1.2.68
confirmation de clé
<sécurité de l'information> assurance pour une entité qu'une autre entité identifiée est en possession de la clé correcte

[SOURCE: ISO/IEC 11770-1:2010, 2.16]

3.1.2.69
retrait d'agrément de clé
<sécurité de l'information> marquage de tous les enregistrements et associations de support de codage pour indiquer que la clé n'est plus utilisée

3.1.2.70
dérivation de clé
<sécurité de l'information> processus par lequel une ou plusieurs clés sont dérivées d'une information secrète partagée et autres informations

3.1.2.71
distribution de clés
<sécurité de l'information> transport d'une clé et autre support de codage d'une entité qui soit est propriétaire de la clé, soit génère la clé, vers une autre entité qui a l'intention d'utiliser la clé

3.1.2.72
centre de distribution de clés
entité qui est habilitée à générer ou acquérir des clés et les distribuer à des parties en communication et qui partage une clé symétrique unique avec chacune des parties

[SOURCE: ISO/IEC 11770-1:2010, 2.22]

3.1.2.73
clé de chiffrement de clé
<sécurité de l'information> clé cryptographique qui est utilisée pour le chiffrement ou le déchiffrement d'autres clés

Note 1 à l'article: Les bonnes pratiques visent à limiter l'utilisation des KEK (secrètes) symétriques à l'enveloppement de clés et de ne pas les utiliser pour le transport de clés ou comme clés de session (à savoir des données).

Note 2 à l'article: Des KEK peuvent former une hiérarchie. Dans la présente norme, les KEK sont souvent appelées "clés principales", bien que les deux concepts ne soient pas synonymes.

3.1.2.74
dépôt de clé
<sécurité de l'information> processus d'enregistrement de clés et des informations de récupération de clé essentielles associées dans une archive de clés

3.1.2.75
établissement de clé
<sécurité de l'information> processus par lequel des clés cryptographiques sont distribuées en toute sécurité parmi des modules cryptographiques utilisant des méthodes manuelles de transport (chargeurs de clés, par exemple), des méthodes automatisées (protocoles de transport de clés et/ou d'agrément de clés, par exemple) ou une combinaison de méthodes automatisées et manuelles (constituée du transport de clés plus l'agrément de clé)

3.1.2.76
installation de support de codage
<sécurité de l'information> installation d'un support de codage pour une utilisation opérationnelle

3.1.2.77
gestion de clés

<sécurité de l'information> administration et utilisation de la production, de l'enregistrement, de la certification, du retrait d'agrément, de la distribution, de l'installation, du stockage, de l'archivage, de la révocation, de la dérivation et de la destruction du support de codage conformément à une politique de sécurité

[SOURCE: ISO/IEC 11770-1:2010, 2.28]

3.1.2.78
infrastructure de gestion de clé
<sécurité de l'information> cadre et services qui permettent la génération, la production, la distribution, la commande, la comptabilisation et la destruction de tout le support cryptographique, y compris les clés symétriques, ainsi que la signature à clé publique et la génération de ses propres paires de clés asymétriques statiques et éphémères

Note 1 à l'article: Y sont inclus tous les éléments (matériels, logiciels, autres équipements et documentation); installations; personnel; procédures; normes; et produits informationnels qui forment le système qui distribue, gère et prend en charge la livraison de produits et de services cryptographiques à des utilisateurs finals.

Note 2 à l'article: Les services de gestion de clés comprennent la commande de clés, la distribution, le recodage, la mise à jour des attributs du support de codage, la révocation de certificat, la récupération de clé et la distribution, la comptabilisation, le suivi et le contrôle de logiciels qui accomplissent soit des fonctions de sécurité du support de codage, soit des fonctions cryptographiques.

3.1.2.79
gestionnaire de clé
<sécurité de l'information> appareil de l'infrastructure de gestion de clés qui fournit des services de gestion de clés

3.1.2.80
paire de clés
paire de clés asymétriques
<sécurité de l'information> paire de clés correspondantes où la clé privée définit la transformation privée et la clé publique définit la transformation publique

Note 1 à l'article: Une paire de clés est utilisée avec des algorithmes cryptographiques à clés asymétriques.

[SOURCE: ISO/IEC 11770-1:2010, 2.2]

3.1.2.81
récupération de clé
<sécurité de l'information> mécanismes et processus qui permettent à des entités autorisées de récupérer du support de codage dans un stockage sécurisé d'archivage ou de sauvegarde de clés

3.1.2.82
enregistrement de clé
<sécurité de l'information> processus d'enregistrement officiel du support de codage par une autorité d'enregistrement

3.1.2.83
révocation de clé
<sécurité de l'information> processus par lequel est rendu disponible à des entités affectées un avis qu'il convient de retirer le support de codage de l'utilisation opérationnelle avant la fin de la période cryptographique établie de ce support de codage

3.1.2.84
transport de clé
<sécurité de l'information> processus de transfert d'une clé d'une entité vers une autre entité, convenablement protégée

Note 1 à l'article: Lorsqu'il est utilisé conjointement à un algorithme (asymétrique) à clé publique, le support de codage est chiffré à l'aide de la clé publique du destinataire et ensuite déchiffré à l'aide de la clé privée du destinataire. Lorsqu'il est utilisé conjointement à un algorithme symétrique, le support de codage est enveloppé avec une clé de chiffrement de clé partagée par les deux parties prenantes.

[SOURCE: ISO/IEC 11770-1:2010, 2.33]

3.1.2.85
mise à jour de clé
<sécurité de l'information> fonction accomplie sur une clé cryptographique afin de calculer une clé nouvelle, mais correspondante

3.1.2.86
période d'utilisation de clé
<sécurité de l'information> période d'utilisation de l'émetteur ou période d'utilisation de destinataire d'une clé symétrique

3.1.2.87
enveloppement de clé
<sécurité de l'information> méthode de chiffrement de clés (avec les informations d'intégrité associées) qui fournit une protection tant de la confidentialité que de l'intégrité en utilisant une clé symétrique

3.1.2.88
clé d'enveloppement de clé
<sécurité de l'information> clé de chiffrement de clé (à clé symétrique)

3.1.2.89
support de codage
<sécurité de l'information> données nécessaire pour établir et maintenir des relations de codage cryptographiques

EXEMPLE Clés, valeurs d'initialisation, périodes de validité.

[SOURCE: ISO/IEC 11770-1:2010, 2.27]

3.1.2.90
latence
retard entre le moment où la donnée est créée à un appareil source de données et le moment où elle est disponible pour être consommée à l'appareil de destination

Note 1 à l'article: Les points de mesure désignés sont a) les appareils physiques, ou b) les frontières entre couches au sein d'un logiciel à couches multiples (par exemple, fonctionnalité d'un transport d'envoi à un transport de réception, ou d'une application émettrice vers un modem expéditeur).

3.1.2.91
location
allocation de ressources de communication à grains fins par session apparaissant en un GIAP

3.1.2.92
moindre privilège
<sécurité de l'information> principe de sécurité qui restreint les privilèges d'accès (privilèges d'exécution de programme, privilèges de modification de fichiers, par exemple) du personnel autorisé et de ses cyberagents au minimum nécessaire pour accomplir leurs tâches

3.1.2.93
liaison
chemin d'interconnexion, momentané ou persistant, entre au moins deux appareils à des fins d'émission et de réception de messagerie

3.1.2.94
clé principale
<sécurité de l'information> clé cryptographique qui est utilisée pour dériver d'autres clés

Note 1 à l'article: Les bonnes pratiques interdisent d'utiliser les clés principales comme clés de session (c'est-à-dire de données), ce qui faciliterait leur analyse cryptographique. Elles peuvent être utilisées comme des KEK, souvent au sommet de la hiérarchie des KEK.

3.1.2.95
topologie maillée
topologie de réseau dans laquelle des chemins redondants d'acheminement physiquement différents sont disponibles entre chaque paire de nœuds du réseau

Note 1 à l'article: La topologie maillée sans fil est utilisable pour étendre la couverture par une fonctionnalité de sauts multiples et/ou pour faciliter la fiabilité des communications par la fourniture de chemins redondants entre les appareils.

3.1.2.96
authentification de message
authentification de PDU
<sécurité de l'information> processus établissant qu'un message a été formé par un membre d'un groupe autorisé de communicants et que le message est inchangé depuis qu'il a été formé

3.1.2.97
code d'authentification de message
code d'intégrité de message
<sécurité de l'information> somme de contrôle cryptographique généré à l'aide d'une clé symétrique qui est typiquement aboutée à des données afin d'assurer une intégrité des données et une authentification de la source semblables à une signature numérique

[SOURCE: ISO/IEC 26907:2009, 4.16]

3.1.2.98
algorithme de code d'authentification de message
<sécurité de l'information> algorithme de calcul d'une fonction qui projette des chaînes de bits et une clé secrète sur des chaînes de bits de taille fixe, en satisfaisant aux deux propriétés suivantes:

- pour toute clé et toute chaîne d'entrée, la fonction peut être calculée efficacement;
- pour toute clé fixe, et sans connaissance préalable de la clé, il est irréalisable par le calcul de calculer la valeur de la fonction sur toute nouvelle chaîne d'entrée, même en ayant connaissance d'un ensemble de chaînes d'entrée et de valeurs de fonction correspondantes, la valeur de chaîne d'entrée de rang i pouvant avoir été choisi après avoir observé la valeur des premières i-1 valeurs de fonction (pour des nombres entiers $i > 1$)

[SOURCE: ISO/IEC 9797-1:2011, 3.10, modifiée – suppression des notes jugées non appropriées pour la présente norme.]

3.1.2.99
syntaxe du calcul de MIC
<sécurité de l'information> représentation concrète d'une information de protocole associée à une N-SDU, habituellement ajoutée au moyen d'un pseudo-en-tête de préfixe, qui est utilisée pour lier des N-addresses sélectives et des N-PCI et parfois des (N-1)-addresses sélectives et des (N-1)-PCI à la N-SDU avant de calculer un code de contrôle d'intégrité (MIC) sur l'assemblage

3.1.2.100
multidiffusion
messagerie d'une source vers un ensemble de destinataires prévus

Note 1 à l'article: L'appartenance comme membre de l'ensemble est indéterminée ou déterminée, le dernier cas incluant l'ensemble vide.

Note 2 à l'article: La diffusion est une forme spéciale de la multidiffusion, habituellement vers un ensemble indéterminé de destinataires prévus. Voir également monodiffusion.

Note 3 à l'article: La multidiffusion, autre que la diffusion, n'est pas prise en charge dans la présente norme.

3.1.2.101
objet de gestion de réseau
logiciel d'application au sein d'un appareil qui agit comme homologue local d'un gestionnaire de réseau

3.1.2.102
générateur non déterministe de bits
<sécurité de l'information> générateur aléatoire de bits dont la sécurité dépend de l'échantillonnage d'une source d'entropie

Note 1 à l'article: Les sources de tels bits incluent le claquage par avalanche d'une diode Zener, le bruit de grenaille, le bruit thermique, la décroissance/désintégration radioactives, les rayons cosmiques, etc.

Note 2 à l'article: Le post-traitement de telles sources de bruit est exigé pour blanchir leur sortie et pour détecter des défaillances dans le circuit fournissant les nombres aléatoires. Seul le train post-traité de bits est adapté pour l'ensemencement d'un générateur déterministe de bits.

[SOURCE: ISO/IEC 18031:2011, 3.23, modifiée – la note à l'article initiale a été supprimée, et de nouvelles notes à l'article ont été ajoutées.]

3.1.2.103
non-répudiation
<sécurité de l'information> capacité à prouver l'occurrence d'un événement ou d'une action donné et les entités qui en sont à l'origine, de manière résoudre les litiges entre l'occurrence ou la non-occurrence de l'événement ou de l'action et l'implication des entités dans l'événement

Note 1 à l'article: Dans un contexte général de sécurité de l'information, la non-répudiation donne l'assurance que l'émetteur de l'information est pourvu d'une preuve durable de la livraison ou le destinataire est pourvu d'une preuve durable de l'identité de l'émetteur, si bien que la partie prenante qui a fourni la preuve irréfutable n'a aucun démenti ultérieur crédible d'avoir traité l'information.

[SOURCE: ISO/IEC 27000:2014, 2.54, modifiée – l'article d'intention "afin de…" a été ajouté.]

3.1.2.104
nonce
<sécurité de l'information> nombre utilisé une seule fois ou valeur qui a (tout au plus) une chance négligeable de se répéter

3.1.2.105
phase opérationnelle
utilisation opérationnelle
<sécurité de l'information> phase dans le cycle de vie d'un support de codage par laquelle le support de codage est utilisé pour des besoins cryptographiques normalisés

3.1.2.106
stockage opérationnel
<sécurité de l'information> stockage normal du support de codage opérationnel pendant sa période cryptographique

3.1.2.107
période d'usage d'émetteur
<sécurité de l'information> durée pendant la période cryptographique d'une clé symétrique pendant laquelle une protection cryptographique peut être appliquée à des données

3.1.2.108
durée de protection
<sécurité de l'information> durée pendant laquelle l'intégrité et/ou la confidentialité d'une clé a/ont besoin d'être maintenue(s)

3.1.2.109
texte clair
<sécurité de l'information> information non chiffrée (relative à un processus de chiffrement ou de déchiffrement)

Note 1 à l'article: Habituellement, le texte clair injecté dans une opération de chiffrement n'est pas déjà chiffré, mais dans certain cas, l'entrée est elle-même la sortie d'une autre opération de cryptographique.

[SOURCE: ISO/IEC 10116:2006, 3.11, modifiée par le commentaire entre parenthèses.]

3.1.2.110
gestion politique
approche administrative (de gestion) utilisée pour simplifier la gestion d'un système donné par l'établissement de politiques dans le but de traiter de situations qui sont comprises comme étant susceptibles de se produire

3.1.2.111
clé privée
<sécurité de l'information> clé (cryptographique) d'une paire de clés asymétriques d'une entité qui est maintenue privée

Note 1 à l'article: La sécurité d'un système asymétrique dépend de la confidentialité de cette clé.

Note 2 à l'article: Dans un système cryptographique (public) asymétrique, la clé privée est associée à une clé publique. La clé privée n'est connue que du propriétaire de la paire de clés et sert à:

– calculer la clé publique correspondante;
– calculer une signature numérique qui est vérifiable par la clé publique correspondante;
– déchiffrer des données qui avaient été chiffrées par la clé publique correspondante; ou
– calculer un élément de données partagées communes, avec d'autres informations.

[SOURCE: ISO/IEC 11770-1:2010, 2.35, modifiée – une seconde note à l'article a été ajoutée]

3.1.2.112
pseudo-en-tête
information qui est logiquement ajoutée au début d'une PDU avant le calcul d'un MIC pour la PDU, mais qui n'est pas explicitement acheminée par la PDU

3.1.2.113
clé publique
clé d'une paire de clés asymétriques d'une entité qui peut être habituellement rendue publique sans compromettre la sécurité

[SOURCE: ISO/IEC 11770-1:2010, 2.36]

3.1.2.114
certificat de clé publique
<sécurité de l'information> information de clé publique d'une entité signée par l'autorité de certification

Note 1 à l'article: Des informations complémentaires contenues dans le certificat sont capables de spécifier comment la clé est utilisée et sa période cryptographique.

[SOURCE: ISO/IEC 11770-1:2010, 2.37, modifiée – une note à l'article a été ajoutée.]

3.1.2.115
période d'usage de destinataire
<sécurité de l'information> période de temps dans une période cryptographique d'une clé symétrique pendant laquelle les informations protégées sont traitées

Note 1 à l'article: Cette période se prolonge fréquemment au-delà de la période d'usage permis de l'émetteur.

3.1.2.116
résilience
aptitude d'une unité fonctionnelle à continuer d'accomplir une fonction exigée en présence d'anomalies ou d'erreurs

[SOURCE: ISO/IEC 2382-14:1997, 14.04.06]

3.1.2.117
période de rétention
<sécurité de l'information> durée minimale pendant laquelle il convient qu'une clé ou autre information cryptographique connexe soit conservée dans une archive

3.1.2.118
robustesse
degré auquel un système ou un composant peut fonctionner correctement en présence de données d'entrée non valides ou de conditions d'environnement stressantes

[SOURCE: ISO/IEC/IEEE 24765:2010, 3.2601]

3.1.2.119
routeur
appareil qui transmet des NPDU au sein d'un réseau informatique basées sur des informations de couche réseau

3.1.2.120
signalisation SCS
message MAC court envoyé par un destinataire nominal de message MAC en réponse immédiate à l'émetteur de ce message MAC et utilisé pour la transmission des informations de contrôle (état ARQ ACK/NAK, qualité et niveau du signal reçu, etc.) dans le but d'informer l'appareil émetteur de l'état de réception et des conditions instantanées sur le support

Note 1 à l'article: L'abréviation SCS est dérivée du terme anglais développé correspondant "short control signaling".

3.1.2.121
clé secrète
<sécurité de l'information> clé utilisée avec des techniques cryptographiques symétriques par un ensemble spécifié d'entités

[SOURCE: ISO/IEC 11770-3:2008, 3.35]

3.1.2.122
protocole de communications sécurisé
<sécurité de l'information> protocole de communication qui fournit la confidentialité, l'authentification, l'intégrité de contenus et la protection de temporisation de message appropriées

3.1.2.123
association de sécurité
<sécurité de l'information> relation entre deux entités ou plus pour lesquelles il existe des attributs (règles et informations d'état) régissant la fourniture des services de sécurité qui intéressent les entités en question

[SOURCE: ISO/IEC 10745:1995, 3.8]

3.1.2.124
domaine de sécurité
<sécurité de l'information> ensemble de biens et de ressources associé à une politique de sécurité commune

Note 1 à l'article: Les domaines de sécurité sont souvent organisés (hiérarchiquement, par exemple) pour former des domaines plus vastes.

[SOURCE: ISO/IEC 18028-3:2005, 3.19, modifiée – une note à l'article a été ajoutée]

3.1.2.125
gestionnaire de sécurité
<sécurité de l'information> logiciel d'application qui supervise divers aspects de sécurité opérationnels d'un réseau de plusieurs appareils, habituellement par le biais de l'interaction avec des objets de gestion de la sécurité d'appareils (DSMO) dans le ou les appareils supervisés

Note 1 à l'article: Un gestionnaire de sécurité réseau est souvent un appareil spécialisé qui est protégé tant physiquement que par sa construction. Voir ISO/IEC 19790 (similaire à FIPS 140-2) et le programme de validation des modules cryptographiques NIST/CSE, http://csrc.nist.gov/cryptval/cmvp.htm.

3.1.2.126
services de sécurité
<sécurité de l'information> mécanismes utilisés pour fournir la confidentialité, l'intégrité des données, l'authentification et/ou la non-répudiation des informations

3.1.2.127
séparation des tâches
<sécurité de l'information> principe de sécurité qui divise des fonctions critiques entre différents membres du personnel dans une tentative d'assurer qu'aucun individu pris séparément ne possède suffisamment d'informations ou de privilèges d'accès pour perpétrer une fraude dommageable

3.1.2.128
session
association T

3.1.2.129
clé de la session
clé T
clé de donnée temporaire utilisée par des TLE

3.1.2.130
production de signature
<sécurité de l'information> utilisation d'un algorithme de signature numérique et d'une clé privée pour créer une signature numérique sur des données

3.1.2.131
authentification de signature
<sécurité de l'information> utilisation d'un algorithme de signature numérique et d'une clé publique pour vérifier une signature numérique sur des données

3.1.2.132
authentification de source
<sécurité de l'information> processus de confirmation que la source des données est telle que déclarée

3.1.2.133
connaissance répartie
<sécurité de l'information> processus par lequel une clé cryptographique est répartie en plusieurs composantes de clés, qui, isolément, ne divulguent aucune connaissance de la clé d'origine autre que sa taille éventuellement et qui peuvent ensuite être combinées en un nombre quelconque prédéfini de groupes des plusieurs clés spécifiques pour récréer la clé d'origine

3.1.2.134
clé statique
<sécurité de l'information> clé qui n'est pas une clé éphémère, qui est prévue pour être utilisée pendant une durée relativement longue, typiquement dans des invocations successives d'un plan d'établissement de clés cryptographiques

3.1.2.135
chiffre continu
<sécurité de l'information> primitive cryptographique qui utilise une clé symétrique et un vecteur d'initialisation et qui fonctionne avec des flux continus d'entrée plutôt qu'avec des blocs fixes

3.1.2.136
sous-réseau
sous-réseau (N)
sous-réseau D
sous-réseau N
sous-réseau, sous-ensemble d'un réseau complet, soit en une couche liaison de données, soit en couche réseau (couche 2 ou 3 du Modèle de référence de base OSI), constitué de plusieurs nœuds de point d'extrémité et nœuds relais qui sont interconnectés via une couche (N-1) fréquemment homogène

3.1.2.137
supertrame
ensemble d'intervalles de temps ayant une période de répétition commune et éventuellement d'autres attributs communs

3.1.2.138
clé symétrique
<sécurité de l'information> clé secrète partagée entre deux parties ou plus qui peut être utilisée tant pour le chiffrement que le déchiffrement ainsi que pour le calcul et la vérification du code d'intégrité de message

[SOURCE: ISO/IEC 26907:2009, 4.27]

3.1.2.139
algorithme à clés symétriques
algorithme (cryptographique) à clés symétriques
<sécurité de l'information> algorithme cryptographique qui utilise la même clé (habituellement secrète) pour une opération et son inverse (par exemple: chiffrement et déchiffrement)

3.1.2.140
initialisation de système
<sécurité de l'information> établissement et configuration d'un système pour un fonctionnement sécurisé

3.1.2.141
gestionnaire de système
logiciel d'application qui supervise divers aspects opérationnels d'un réseau de plusieurs appareils autres que la sécurité, habituellement par le biais de l'interaction avec des objets de gestion de réseau dans l'(les) appareil(s) supervisé(s)

Note 1 à l'article: Un gestionnaire de réseau soit supervise un réseau entier à plusieurs appareils, soit agit comme un subordonné d'un autre gestionnaire de réseau, ne supervisant de ce fait qu'un sous-ensemble du réseau entier.

Note 2 à l'article: Un gestionnaire de réseau n'est pas toujours un appareil spécialisé.

3.1.2.142
gestionnaire de sécurité/système
fonctionnalité de gestionnaire de système agissant pour le compte de la fonctionnalité de gestionnaire de sécurité

3.1.2.143
menace
<sécurité de l'information> circonstance ou événement ayant le potentiel d'avoir un impact défavorable sur les opérations de l'installation (y compris le fonctionnement, l'image ou la réputation), les biens ou les individus à travers un système d'information au moyen d'un accès non autorisé, d'une destruction, d'une divulgation, d'une modification de données, d'un retard de message et/ou d'un déni de service

3.1.2.144
seuil
valeur qui, lorsqu'elle est dépassée dans un sens désigné, engendre une réaction spécifique

3.1.2.145
fenêtre de temps
intervalle de temps

3.1.2.146
transaction D
MPDU qui n'est pas une immédiate MPDU d'acquittement, plus toute séquence de zéro à plusieurs immédiates MPDU d'acquittement qui suivent immédiatement et sont une conséquence de la première MPDU (et éventuellement les unes des autres), toutes sur la même voie et dans le même intervalle de temps

3.1.2.147
tunnellisation
encapsulation d'un premier protocole dans un second protocole de communication pour acheminer des PDU issues du premier protocole

3.1.2.148
type de service
nom collectif donné à un ensemble d'éléments de protocoles et d'attributs de qualité de service associés qui forment ensemble un sous-protocole (par exemple: voix temps réel, données en temps critique et données en temps non critique) avec une fonctionnalité distincte

3.1.2.149
divulgation non autorisée
<sécurité de l'information> événement impliquant l'exposition d'informations à des entités interdites d'accès aux informations

3.1.2.150
monodiffusion
messagerie d'une source vers un seul destinataire prévu

Note 1 à l'article: Voir aussi multidiffusion et diffusion.

3.1.2.151
initialisation d'utilisateur
processus par lequel un utilisateur établit l'application cryptographique en vue de son utilisation (par exemple: installation et configuration de logiciel et de matériel)

3.1.2.152
enregistrement d'utilisateur
processus par lequel une entité devient membre d'un domaine de sécurité

3.1.2.153
abrogation
<sécurité de l'information> méthode permettant par un procédé électronique l'effacement de données mémorisées, de clés cryptographiques et de paramètres mémorisés critiques en

altérant ou supprimant le contenu du stockage d'une manière qui empêche la récupération des données

3.1.3 Symboles pour les clés symétriques, les clés asymétriques et les certificats

NOTE Les clés symétriques définies par la présente norme sont des clés de 128 bits. Les clés asymétriques définies par la présente norme ont une force de bit cryptographique hypothétique similaire de 128 bits ou plus. Voir Article 7 pour la description exacte du support cryptographique.

3.1.3.1
K_DL
clé de données courante pour tous les appareils du sous-réseau D local

3.1.3.2
K_global
clé bien connue dont la valeur est statique et éditée, utilisée pour assurer l'uniformité de la structure et du traitement des PDU lorsqu'une clé secrète partagée est inappropriée ou inconnue

3.1.3.3
K_open
clé d'utilité limitée bien connue, différente de K_global, dont la valeur est statique et publiée, qui peut être utilisée comme valeur de K_join pour configurer un appareil via son interface de liaison radio

Note 1 à l'article: L'utilisation de cette clé dans un environnement où de l'écoute illicite pourrait survenir peut compromettre la sécurité de l'appareil ainsi que ses relations avec le reste du système sans fil.

3.1.3.4
K_join
clé utilisée pour amorcer de manière sécurisée un nouvel appareil dans le réseau

3.1.3.5
K_join_wrapped
version prête à l'emploi de la clé de rattachement, utilisée pour récupérer d'un gestionnaire de sécurité défaillant

3.1.3.6
K_master
clé principale utilisée comme une KEK pour la distribution de clés et la gestion de sécurité d'un seul appareil

3.1.3.7
K_session_AB
clé de données courante pour une session entre un appareil A et un appareil B, identique avec une K_session_BA

3.1.3.8
CA_root
clé publique d'une autorité de certification qui a signé un certificat de clé publique d'un appareil

Note 1 à l'article: Cette clé est communément appelée une clé racine et sert à aider à la vérification de la véritable identité de l'appareil communiquant le certificat, ainsi qu'un certain nombre d'informations de codage connexes.

3.1.3.9
Cert-A
certificat de clés publiques de l'appareil A, utilisé pour prouver la véritable identité de l'appareil, ainsi que des informations de codage connexes, pendant l'exécution d'un protocole d'établissement de clés à clés publiques authentifiées

3.1.4 Termes utilisés pour décrire le comportement d'un appareil

3.1.4.1
capacité
aptitude à exécuter des actions, comprenant les attributs sur les qualifications et les mesures de cette aptitude en tant que capacité

EXEMPLE Nombre d'appareils connectés qu'un routeur peut prendre en charge

Note 1 à l'article: Les profils spécifient une capacité minimale.

[SOURCE: ISO 18435-2:2012, 3.6, modifié – addition de l'example et de la note]

3.1.4.2
configuration
jeu de paramètres qui 1) altèrent le comportement et 2) peuvent être réglés par un gestionnaire de système

EXEMPLE Limite de sauts de couche réseau.

Note 1 à l'article: Les configurations où des valeurs par défaut sont appropriées énoncent ces valeurs par défaut (par exemple, limite de sauts de NL = 64).

3.1.4.3
configurer
acte consistant à spécifier et gérer les paramètres de configuration

3.1.4.4
caractéristique
caractéristique notable d'un appareil

EXEMPLE Alimenté par pile.

3.1.4.5
obligatoire
élément exigé pour toute revendication de conformité à la présente norme

EXEMPLE Prise en charge pour la cryptographie à clé symétrique.

3.1.4.6
facultatif
élément qui n'est pas exigé pour revendiquer la conformité à la présente norme, mais qui, s'il est présent, doit se comporter comme spécifié dans la présente norme

EXEMPLE Prise en charge pour la cryptographie à clés asymétriques.

3.2 Abréviations et acronymes

6LoWPAN	IPv6 over a low power personal area network (PAN)
6TSCH	time-slotted channel hopping (TSCH) with 6lowpan
ACK	positive acknowledgement (acquittement positif)
AE	application-entity (entité d'application)
AES	advanced encryption standard (norme de chiffrement évolué) (voir ISO/IEC 18033-3)
AID	attribute ID (ID attribut)
AL	application layer (couche d'application)
ALE	application layer entity (entité de couche d'application)
AME	application layer management entity (entité de gestion de couche d'application)
APDU	application layer protocol data unit (unité de données de protocole de couche d'application)
ARMO	alert reporting management object (objet de gestion des rapports d'alertes)
ARO	alert receiving object (objet récepteur d'alerte)

ARQ	automatic repeat request (demande de répétition automatique)
ASAP	application layer service access point (point d'accès aux services de couche d'application)
ASDU	application layer service data unit (unité de données de service de couche d'application)
ASL	application sublayer (sous-couche d'application)
ASLMO	application sublayer management object (objet de gestion de sous-couche d'application)
ASMSAP	application sublayer management service access point (point d'accès au service de gestion de sous-couche d'application)
ASM	asset management (gestion de biens)
ASN.1	abstract syntax notation one (notation de syntaxe abstraite numéro un)
ATT	address translation table (table de conversion d'adresse)
AUTO	automatic mode (mode automatique d'un processus d'automation)
BBR	backbone router (routeur dorsal)
BECN	backward explicit congestion notification (notification explicite d'encombrement vers l'arrière)
BRPT	backbone router peer table (table d'homologues de routeur dorsal)
C/S	client/server (client/serveur)
CBC	cipher-block-chaining stream cipher mode (mode de chiffrement continu par chaÎnage de blocs chiffrants) (voir NIST SP 800-38c)
CCA	clear channel assessment (evaluation de voie libre)
CCM	counter with cipher-block-chaining message authentication mode (mode compteur avec authentification de message par chainage de blocs chiffrants)
CCM*	CCM enhanced (CCM évolué)
CE	Communaute européenne
CIP	Common Industrial Protocol™ [4] (protocole industriel commun)
CON	concentrator object (objet concentrateur)
CoS	class of service (classe de service)
CoSt	change of state (changement d'état)
CSMA	carrier sense multiple access (accès multiple a detection de porteuse)
CSMA/CA	carrier sense multiple access with collision avoidance (accès multiple par détection de porteuse et évitement de collision)
dBm	dB (1 mW)
DADDR	data-link address (adresse de liaison de données)
DAUX	data-link auxiliary header (en-tête auxiliaire de liaison de données)
DBP	device being provisioned (appareil configuré)
DCS	distributed control system (système de contrôle distribué)
DD	device description (description d'appareil)
DDE	data link layer data (sub-)entity ((sous-)entité de données de couche liaison de données)
DDL	device description language (langage de description d'appareil)
DDSAP	data-link layer data service access point (point d'accès au service de données de couche liaison de données)
DE	discard eligible (éligible au rejet)
DIS	dispersion object (objet de dispersion)
DK	(symmetric) data key, data authentication key, data encryption key (clé de données (symétrique), clé d'authentification de données, clé de chiffrement de données)
DL	data-link layer (couche liaison de données)
DLE	data-link layer entity (entité de couche liaison de données)
DLMO	data-link layer management object (objet de gestion de couche liaison de données)
DMAP	device management application process (processus d'application de gestion d'appareil)
DME	data link layer management (sub-)entity ((sous-)entité de gestion de couche liaison de données)

[4] Propriété d'ODVA, http://www.odva.org.

DMIC	data-link layer message integrity code (code d'intégrité de message de couche liaison de données)
DMO	device management object (objet de gestion d'appareil)
DMSAP	data-link layer management service access point (point d'accès au service de gestion de couche liaison de données)
DMSO	device management service object (objet de service de gestion d'appareil)
DMXHR	data-link layer management extension header (en-tête d'extension de gestion de couche liaison de données)
DPDU	data-link layer protocol data unit (unité de données de protocole de couche liaison de données)
DPO	device provisioning object (objet de configuration d'appareil)
DPSO	device provisioning service object (objet service de configuration d'appareil)
DROUT	data-link layer routing (subheader) (sous-en-tête) de routage de couche liaison de donnees
DSAP	data-link layer service access point (point d'accès au service de couche liaison de donnees)
DSC	data-link layer security component (composant de sécurité de couche liaison de données)
DSDU	data-link layer service data unit (unité de données de service de couche liaison de données)
DSMO	device security management object (objet de gestion de sécurité d'appareil)
DSO	directory service object (objet de service d'annuaire)
DSSS	direct sequence spread spectrum (étalement du spectre en séquence directe)
DWT	data-link layer protocol data unit wait time (durée d'attente d'unité de données de protocole de couche liaison de données)
ECB	electronic code book (livre de codes électroniques)
ECC	elliptic curve cryptography standard (norme de cryptographie sur courbes elliptiques) (voir ISO/IEC 18033-2)
ECMQV	Menezes-Qu-Vanstone algorithms using elliptic curve cryptography (algorithmes de Menezes-Qu-Vanstone utilisant la cryptographie sur courbes elliptiques)
ECN	explicit congestion notification (notification d'encombrement explicite)
ECPVS	Pintsov-Vanstone algorithms using elliptic curve cryptography (algorithmes de Pintsov-Vanstone utilisant la cryptographie sur courbes elliptiques)
ECQV	Qu-Vanstone algorithms using elliptic curve cryptography (algorithmes de Qu-Vanstone utilisant la cryptographie sur courbes elliptiques)
ED	energy detection (détection d'énergie)
EDDL	electronic device description language (langage de description électronique de produit)
EIRP	effective isotropic radiated power (puissance rayonnée isotropique réelle)
EMA	exponential moving average (moyenne mobile exponentielle)
ETSI	European Telecommunications Standards Institute (Institut Européen des Normes de Telecommunication)
EUI-64™[5]	identificateur unique etendu de 64 bits tel que spécifié par l'IEEE
FDT/DTM	field device tool/device type manager (outil d'appareil de terrain/gestionnaire de type d'appareil)
FCC	[US] Federal Communications Commission (agence fédérale américaine de régulation des communications)
FCS	frame check sequence (séquence de contrôle de trame)
FEC	forward error correction (correction d'erreur directe)
FECN	forward explicit congestion notification (notification explicite d'encombrement vers l'avant)
FF-H1	Foundation Fieldbus – H1 protocol (Fondation Fieldbus -protocole h1)
FF-HSE	Foundation Fieldbus – high speed ethernet (Fondation Fieldbus – ethernet haut débit)
FHSS	frequency hopping spread spectrum (étalement de spectre à saut de fréquence)

[5] Propriété du détenteur de la marque.

FHSSM	frequency hopping spread spectrum modulation (modulation à étalement de spectre à saut de fréquence)
FIFO	first in, first out (premier entré, premier sorti – discipline de mise en file d'attente)
FIPS	[US] federal information processing standard (norme fédérale de traitement de l'information, produite par le NIST (Institut national américain des sciences et de la technologie))
FPAC	foreign protocol application communication (communication d'application de protocole étranger)
FPT	foreign protocol translator (convertisseur de protocole étranger)
FPT-PAI	foreign protocol translator – protocol address information (convertisseur de protocole étranger – informations d'adresse de protocole)
FPT-PCI	foreign protocol translator – protocol control information (convertisseur de protocole étranger – informations de controle de protocole)
FPT-PDU	foreign protocol translator – protocol data unit (convertisseur de protocole étranger – unité de données de protocole)
MDF	frequency shift keying (modulation par déplacement de fréquence)
GIAP	gateway interface access point (point d'accès à interface de passerelle)
GPS	global positioning system (système de positionnement global)
GUC	gateway-UAP connections (nombre de connexions gateway-UAP prises en charge)
GUI	graphical user interface (interface utilisateur graphique)
HART	highway addressable remote transducer (transducteur distant adressable par voie express)
HC	header compression (compression d'en-tête)
HCF	Hart Communication Foundation (Fondation de communications Hart)
HMAC	(keyed)-hash message authentication code (code d'authentification de message à fonction de hachage codée)
IHM	human-machine interface (interface homme-machine)
HRCO	health reports concentrator object (objet concentrateur de bulletins de santé)
(N)-ICI	(N-layer) interface control information (informations de contrôle d'interface (couche N))
ICMP	internet control messaging protocol (protocole de messagerie de controle internet)
IEC	International Electrotechnical Commission, Commission Electrotechnique Internationale
IEEE	Institute of Electrical & Electronics Engineers (Institut des ingénieurs électriciens et électroniques)
IFO	interface object (objet interface)
INF	infinity (infini)
E/S	entrée/sortie
IPv4	internet protocol (protocole internet) version 4
IPv6	internet protocol (protocole internet) version 6
ISA	International Society of Automation (Société internationale d'automation)
ISM	industrial, scientific, medical (industriel, scientifique, médical)
IV	initialization vector (vecteur d'initialisation)
JT_n	join timer n (temporisateur de rattachement n)
KDC	key distribution center (centre de distribution de clés)
KEK	key encryption key (clé de chiffrement de clé)
LAN	local area network (réseau local)
LBT	listen before talk (écouter avant de parler)
LH	last hop (dernier saut)
LLC	logical link control sublayer (sous-couche de contrôle de liaison logique, dans la DL supérieure)
LP	low power (faible puissance)
LQI	link quality indicator (indicateur de qualité de liaison)
LSB	least significant bit (bit de poids faible)
MAC	media access control sublayer (sous-couche de contrôle d'accès au support, enjambant la couche dl inférieure et la couche PhL supérieure)
MAN	manual mode (mode manuel d'un processus d'automation)
MIB	management information base (base d'informations de gestion)

MIC	message integrity code (code d'intégrité de message)
MICI	media access control interface control information (information de contrôle d'interface de contrôle d'accès au support)
MK	(symmetric) master key (clé principale (symétrique))
MP	management process (processus de gestion)
MPCI	media access control sublayer protocol control information (informations de contrôle de protocole de sous-couche de contrôle d'accès au support)
MPDU	media access control sublayer protocol data unit (unité de données de protocole de sous-couche de contrôle d'accès au support)
MSB	most significant bit (bit de poids fort)
NAK	negative acknowledgment (acquittement negatif)
NaN	not-a-number (non-nombre)
NDE	network layer data (sub-)entity ((sous-)entité de données de couche réseau)
NDSAP	network layer data service access point (point d'accès au service de données de couche réseau)
NFC	near-field communications (communications de champ proche)
NICI	network interface control information (informations de contrôle d'interface réseau)
NIDS	network intrusion detection system (système de détection des intrusions de réseau)
NIST	[Us] National Institute of Standards And Technology (Institut national des normes et de la technologie)
NL	network layer (couche réseau)
NLE	network layer entity (entité de couche réseau)
NME	network layer management (sub-)entity ((sous-)entité de gestion de couche réseau)
NMSAP	network layer management service access point (point d'accès au service de gestion de couche réseau)
NO	native object (objet natif)
NPCI	network layer protocol control information (informations de contrôle de protocole de couche réseau)
NPDU	network layer protocol data unit (unité de données de protocole de couche réseau)
NPDU.F128	adresse IPv6address de destination finale dans l'en-tête de réseau étendu
NPDU.F16	adresse DL16Address de destination finale dans l'en-tete de réseau étendu
NPDU.O128	adresse IPv6Address d'émetteur dans l'en-tête de réseau étendu
NPDU.O16	adresse DL16Address d'émetteur dans l'en-tête de réseau étendu
NSAP	network layer service access point (point d'accès aux services de couche réseau)
NSD	number of system devices (nombre d'appareils système)
NSDU	network layer service data unit (unité de données de service de couche reséau)
OBJ	object (objet de couche d'application générique)
ODVA	Open Device Vendor Association (Association de fournisseurs d'appareils ouverts); désormais connue légalement sous son acronyme seulement
OOB	out-of-band (hors bande)
OOS	out-of-service mode (mode hors service d'un processus d'automation)
OP	output point (point de sortie d'un processus d'automation)
OPC	open connectivity in industrial automation (connectivité ouverte dans l'automation industrielle)
OSI	Open Systems Interconnection (interconnexion des systèmes ouverts)
OTA	over-the-air (en liaison radio, par voie hertzienne)
P/S	publish/subscribe (éditer/s'abonner)
PA	process automation (automation de processus)
(N)-PAI	(n-layer) protocol addressing information (informations d'adressage de protocole (couche n))
PAN	personal area network (reseau local personnel)
PCH	phy coding header (en-tete de codage phy)
(N)-PCI	(N-layer) protocol control information (informations de contrôle de protocole (couche N))
PD	provisioning device (appareil de configuration)
(N)-PDU	(N-layer) protocol data unit (unite de données de protocole (couche N))

PHD	physical layer data service (service de données de couche physique)
PhICI	physical layer interface control information (informations de contrôle d'interface de couche physique)
PhL	physical layer (couche physique)
PhLE	physical layer entity (entité de couche physique)
PhPDU	physical layer protocol data unit (unité de données de protocole de couche physique)
PhSAP	physical layer service access point (point d'accès aux services de couche physique)
PhSDU	physical layer service data unit (unité de données de service de couche physique)
PHY	couche physique (telle qu'utilisée dans les normes IEEE 802)
PIB	policy information base (base d'informations de politique)
PICS	protocol implementation conformance statement (déclaration de conformité de mise en œuvre d'un protocole)
PKI	public key infrastructure (infrastructure de cle publique)
PNO	PROFIBUS Nutzerorganisation (organisation des utilisateurs PROFIBUS)
PSH	PHY synchronization header (en-tête de synchronisation PHY)
PSMO	proxy security management object (objet proxy de gestion de sécurité)
PV	process variable (variable de processus d'un processus d'automation)
PWT	PDU wait time (temps d'attente de PDU)
QoS	quality of service (qualité de service)
R&TTE (ETRT)	radio and telecommunications terminal equipment (équipement terminal de radio et de télécommunication)
RDP	reliable datagram protocol (protocole datagramme fiable)
RF	radio frequency (radiofréquence)
RFC	request for comments (demande de commentaires)
RFP	request for proposal (demande de proposition)
RSSI	received signal strength indicator (indicateur d'intensité de signal en réception)
RSQI	received signal quality indicator (indicateur de qualité de signal en réception)
RT	routing table (table de routage)
RTO	retry time-out interval (intervalle de temporisation entre essais)
RTT	round-trip time (temps de propagation aller-retour)
RTU	remote terminal unit (unité terminale distante)
RTTV	round-trip time variation (variation de temps de propagation aller-retour)
S/S	source/sink (source/puits)
(N)-SAP	(N-layer) service access point (point d'accès au service (couche N))
SCADA	supervisory control and data acquisition (système de supervision, contrôle et acquisition de données)
SCS	short control signaling (signalisation scs)
SCO	system communication configuration object (objet de configuration de communications systeme)
(N)-SDU	(N-layer) service data unit (unite de données de service (couche N))
SFD	start frame delimiter (délimiteur de début de trame)
SIFS	short inter-frame separation (éeparation inter-trames courte)
(U)SIM	(universal) subscriber identity module (module d'identité (universelle) d'abonné)
SINR	signal to interference plus noise ratio (rapport signal/brouillage plus bruit)
SK	clé T (symétrique) utilisée pour l'authentification et la confidentialité
SKG	secret key generation (generation de cle secrete)
SL	session layer (couche session)
SMIB	structured management information base (base d'informations de gestion structurées)
SMAP	system manager application process (processus d'application de gestionnaire de système)
SM	system manager (gestionnaire de système)
SMAP	system management application process (processus d'application de gestion de système)
SMO	system monitoring object (objet de surveillance de système)
SOE	sequence of events (séquence d'événements)

SRTT	smoothed round-trip time (temps de propagation aller-retour lisse)
STSO	system time service object (objet de service de temps système)
TAI	international atomic time (temps atomique international)
TCP	transmission control protocol (protocole de contrôle de transport)
TDMA	time division multiple access (accès multiple par répartition dans le temps)
TDE	transport layer data (sub-)entity ((sous-)entité de données de couche transport)
TDSAP	transport layer data service access point (point d'accès au service de données de couche transport)
TFRC	TCP-friendly rate control (controle de débit compatible TCP), IETF RFC 5348
TICI	transport interface control information (informations de controle d'interface de transport)
TL	transport layer (couche transport)
TLE	transport layer entity (entité de couche transport)
TME	transport layer management (sub-)entity ((sous-)entité de gestion de couche transport)
TMIB	transport layer management information base (base d'informations de gestion de couche transport)
TMIC	transport layer message integrity code (code d'intégrité de message de couche transport)
TMSAP	transport layer management service access point (point d'accès au service de gestion de couche transport)
ToS	type of service (type de service)
TPCI	transport layer protocol control information (informations de contrôle de protocole de couche transport)
TPDU	transport layer protocol data unit (unité de données de protocole de couche transport)
TSAP	transport layer service access point (point d'accès au service de couche transport)
TSC	transport layer security component (composant de sécurité de couche transport)
TSCH	time-slotted channel hopping (saut de voies en fonction du temps)
TSDU	transport layer service data unit (unité de données de service de couche transport)
TUN	tunnel object (objet tunnel)
TUN-Data	tunnel data (données tunnel)
UAL	upper application layer (couche d'application supérieure)
UAP	processus d'application utilisateur
UAPMO	user application process management object (objet de gestion de processus d'application utilisateur)
UDO	upload/download object (objet téléchargement montant/téléchargement descendant)
UDP	user datagram protocol (protocole datagramme d'utilisateur)
UFO	unified field object (objet champ unifié)
UTC (TUC)	universal coordinated time (temps universel coordonné)
WBM	wideband modulation (modulation en large bande)
WiMAX	worldwide interoperability for microwave access (système d'interopérabilité mondiale pour l'accès hyperfréquence)
WISN	wireless industrial sensor network (réseau de capteurs industriels sans fil)

3.3 Conventions

3.3.1 Interfaces de service

Des parties de la présente norme utilisent les conventions descriptives données dans l'ISO/IEC 10731.

Les primitives de service, utilisées pour représenter les interactions utilisateur de service/fournisseur de service (voir ISO/IEC 10731), acheminent des paramètres qui indiquent les informations disponibles dans l'interaction entre utilisateur et fournisseur.

La présente norme utilise un format de tableau pour décrire les paramètres de composants des primitives de NS (SAP couche N ou SAP d'entité). Les paramètres qui s'appliquent à chaque groupe de primitives de NS sont consignés en tableaux. Chaque tableau comporte jusqu'à six colonnes, contenant le nom du paramètre de service, et une colonne chacune pour les primitives et les sens de transfert de paramètres utilisés par le NS:

- les paramètres d'entrée de la primitive "request";
- les paramètres de sortie de la primitive "indication";
- les paramètres d'entrée de la primitive "response"; et
- les paramètres de sortie de la primitive "confirm".

NOTE 1 Les primitives "request", "indication", "response" et "confirm" sont aussi respectivement appelées primitives "requestor.submit", "acceptor.deliver", "acceptor.submit", et "requestor.deliver" (voir ISO/IEC 10731).

Un paramètre (ou une partie de celui-ci) est énuméré dans chaque rangée de chaque tableau. Dans les colonnes appropriées de la primitive de service, un code est utilisé pour spécifier le type d'usage du paramètre sur la primitive et le sens de paramètres spécifiés dans la colonne:

M: le paramètre est obligatoire pour la primitive;

S: une sélection dans un ensemble défini d'au moins deux paramètres;

U: le paramètre est une option de l'utilisateur et peut ou peut ne pas être fourni, cela dépendant de l'usage dynamique de l'utilisateur de NS. Lorsqu'il n'est pas fourni, une valeur par défaut est supposée pour le paramètre;

C: le paramètre est conditionné à d'autres paramètres ou à l'environnement de l'utilisateur de NS;

(blanc/vide) ou tiret cadratin ("—"): le paramètre n'est jamais présent.

Certaines entrées sont en plus qualifiées par des éléments entre parenthèses. Ceux-ci peuvent être:

- une contrainte spécifique au paramètre:

(=) indique que le paramètre équivaut du point de vue de la sémantique au paramètre dans la primitive de service située immédiatement à sa gauche dans le tableau;

- une indication qu'une certaine note s'applique à l'article:

(n) indique que la note n suivante contient des informations complémentaires relatives au paramètre et à son utilisation. Les notes énumérées avec des lettres sont normatives; les notes numérotées avec des chiffres sont informatives.

Un tableau récapitulatif est fourni pour définir l'utilisation du paramètre dans la primitive. Chaque cellule définit si chaque paramètre est obligatoire, facultatif, interdit ou conditionnel.

L'ordre des paramètres est également défini implicitement de bas en haut.

Des paramètres complexes peuvent également être montrés. Par exemple, une structure peut être obligatoire, mais certains des éléments de la structure peuvent être facultatifs. Les éléments d'une structure sont indentés proportionnellement à leur niveau de hiérarchie.

Les paramètres d'entrée pour les services sont spécifiés pour les primitives de service "request" et "response". Les paramètres de sortie pour les services sont spécifiés pour les primitives de service "indication" et "confirmation".

Les abréviations suivantes sont utilisées dans les tableaux des services:

- Demande demande de service
- Indication indication de service

- Réponse réponse de service
- Confirmation confirmation de service

NOTE 2 Le traitement dans un appareil de situations d'erreur entre couches, telles que les situations où une file d'attente de couche inférieure connaît un dépassement de capacité ou il se produit une temporisation de couche inférieure, relève d'une initiative locale et n'est donc pas abordé par la présente norme.

3.3.2 Cellules de tableau

Pour tous les tableaux, les entrées de tableau qui ne sont ni pertinentes ni spécifiées peuvent contenir un tiret cadratin ("—"). Les entrées de tableau devant être remplies par les fournisseurs peuvent contenir des points de suspension ("...").

3.3.3 Italique

Dans certains cas, les termes génériques utilisés à des fins spécifiques sont mis en italique à leur première occurrence afin d'indiquer au lecteur que des précautions d'interprétation sont suggérées. Il convient que les autres usages de l'italique, tels que décrits en 4.5.5.2.1, soient clairement explicités par leur contexte immédiat.

3.3.4 Gras

Dans certains cas, les paragraphes descriptifs sont précédés par une phrase descriptive récapitulative ou un terme en gras, la mise en gras étant utilisée pour aider le lecteur à mémoriser l'information pour plus tard.

NOTE Cet usage du gras n'est pas indispensable, c'est pourquoi la perte d'une telle distinction consécutivement au procédé de photocopie n'est pas importante.

3.3.5 Déclarations informelles de constantes nommées

D'après l'ASN.1, l'attribution de noms symboliques à des constantes numériques est permise. L'attribution de représentations numériques spécifiques à des constantes nommées d'énumérations est également permise. Dans la présente norme, les deux types de déclarations sont unifiés, lorsqu'ils sont utilisés en tant que déclarations incorporées de choix admis concernant des champs de structures de données, grâce à l'utilisation de la syntaxe suivante:

valeurNumérique ": " texte explicatif

qui vise à être équivalente à la déclaration ASN.1

texte_explicatif "(" valeurNumérique ")"

où le texte explicatif est converti en identificateur alphanumérique en remplaçant les espaces figurant entre les mots (s'il y en a) par des tirets de soulignement et en ajustant les caractères de délimitation éventuels (point, virgule ou point-virgule) figurant après le texte explicatif en fonction du séparateur d'éléments de liste ASN.1 exigé après la parenthèse fermante équivalente de la valeurNumérique.

4 Vue d'ensemble

4.1 Généralités

La présente norme utilise la méthodologie de description de couches de l'OSI (voir Annexe C) pour définir des spécifications de suite de protocoles, en plus des spécifications pour les fonctions de sécurité, de gestion, de passerelle et mise à disposition d'un réseau industriel sans fil. Les couches de protocoles prises en charge sont la couche physique (PhL), la couche liaison de données (DL), la couche réseau (NL), la couche transport (TL) et la couche d'application (AL).

NOTE 1 Bien que la présente norme utilise le concept de couches de protocoles, la conformité à la présente norme ne signifie pas que la partition des mises en œuvre ait un fonctionnement similaire à celui impliqué par ces couches. Les interfaces inter-couches ne sont généralement pas exposées dans un produit et ne sont donc pas adéquates pour les essais de conformité.

Le réseau sans fil défini par la présente norme est constitué d'appareils sans fil servant les classes d'utilisation 1 à 5 (décrites à l'Annexe C) pour les applications non critiques d'appareils fixes, portatifs et mobiles.

Les références à un réseau conforme à la présente norme seront appelées ci-après un réseau de capteurs industriels sans fil (WISN), même lorsque le réseau comporte des actionneurs et d'autres appareils qui ne sont pas logiquement classifiables en tant que capteurs.

La plupart des appareils qui participent à un WISN sont censés mettre en œuvre juste une seule PhLE sans fil et la DLE associée. Cependant, les appareils comportant plusieurs PhLE sans fil et les DLE associées ne sont pas exclus. Par conséquent, la présente norme distingue entre les exigences pour un appareil et les exigences pour une PhLE et la DLE associée, même si les deux sont habituellement jugées comme étant synonymes.

NOTE 2 Il est conseillé au lecteur d'intérioriser cette relation et donc que le terme DLE fasse penser au terme appareil, même si dans certains cas, ils ne sont pas en relation biunivoque (1:1).

4.2 Interopérabilité et problèmes associés

L'IEC TR 62390 fournit des définitions utiles pour les différents niveaux d'interopération. Les trois définitions suivantes, chacune incluant la définition précédente, sont citées telles qu'elles apparaissent dans ce rapport technique.

NOTE 1 Dans l'IEC/TR 62390:2005, la Figure 9 apporte des éclaircissements supplémentaires concernant la relation existant entre ces termes.

NOTE 2 Les notes en 4.2 ne figuraient pas dans l'IEC TR 62390.

a) **Interconnectabilité**: Deux appareils ou plus sont interconnectables s'ils utilisent les mêmes protocoles de communication, interface de communication et accès aux données.

b) **Interfonctionnabilité**: Deux appareils ou plus sont interfonctionnables s'ils peuvent transférer des paramètres entre eux; outre le protocole de communication, l'interface de communication et l'accès aux données, les types de données de paramètres sont les mêmes.

NOTE 3 L'interfonctionnabilité implique l'interconnectabilité.

c) **Interopérabilité**: Deux appareils ou plus sont interopérables s'ils peuvent fonctionner ensemble pour accomplir un rôle spécifique dans un ou plusieurs programmes d'application distribués. Les paramètres et leurs fonctionnalités applicatives s'ajustent à la fois de manière syntaxique et sémantique. L'interopérabilité est accomplie lorsque les appareils prennent en charge des ensembles complémentaires de paramètres et fonctions appartenant au même profil.

NOTE 4 L'interopérabilité implique l'interfonctionnabilité et par conséquent l'interconnectabilité aussi.

4.3 Qualité de service

Afin de prendre en charge plusieurs applications au sein d'un réseau avec des besoins divers, la présente norme prend en charge plusieurs niveaux de qualité de service (QoS). La QoS décrit les paramètres tels que la latence, le débit et la fiabilité. Une ou plusieurs application d'un appareil demande le niveau de QoS nécessaire. Si les ressources nécessaires sont mises à la disposition de l'application demandeuse, le gestionnaire du système alloue ces ressources en réponse à la QoS demandée. Voir Article 6 pour des informations complémentaires.

4.4 Applicabilité à l'échelle mondiale

La présente norme vise à se conformer à des réglementations établies dans toutes les régions du monde; cependant, son acceptabilité dans un contexte de réglementation spécifique n'est pas garantie et doit donc être évaluée. L'Annexe V traite de ce thème, y compris l'utilisation dans la CE sous l'ETSI EN 300 328.

4.5 Architecture réseau

4.5.1 Interfaces

4.5.1.1 Interfaces définies

La présente norme définit des points d'accès au service (SAP) à la frontière supérieure de chaque couche de protocoles pour découpler les spécifications ou les révisions de chacune de ces couches de protocoles par rapport aux autres couches, dans toute la mesure du possible. Par exemple, si une nouvelle PhL est définie et n'exige pas d'apporter des modifications à la DL utilisatrice, elle peut être ajoutée à la spécification avec un impact minimal (voire nul) sur les autres couches de protocoles définies par la présente norme.

Dans la plupart des cas, ces interfaces définies sont internes à une mise en œuvre. A ce titre, elles ne sont pas soumises à des vérifications de normalisation et de conformité, car ces vérifications ne peuvent être appliquées qu'à des interfaces externes d'une unité en essai (à savoir des essais de boîte noire). Par conséquent, ces interfaces internes sont descriptives et informatives, pas normatives. Cependant, les mises en œuvre qui partitionnent les logiciels selon les lignes suggérées par ces interfaces sont censées être plus faciles à maintenir et adapter à de futures révisions de la présente norme.

La conformité à la présente norme s'applique seulement au comportement observable d'une mise en œuvre, y compris la structure et le codage de toute information échangée à des interfaces observables qui sont spécifiées comme telles par la présente norme.

4.5.1.2 Interfaces non définies

Les interfaces suivantes ne sont pas traitées par la présente norme.

- Gestionnaire de système – gestionnaire de sécurité: Le gestionnaire de sécurité et le gestionnaire de système forment deux rôles centraux dans le réseau qui sont étroitement liés. Sachant que le gestionnaire de sécurité et le gestionnaire de système sont si dépendants l'un de l'autre et que le gestionnaire de sécurité communique directement seulement avec le gestionnaire de sécurité, il est attendu que le seul ou les quelques appareils fournissant ces deux rôles seront obtenus auprès d'un seul et même fournisseur, en étant réalisés sous la forme d'un seul appareil prenant en charge les deux rôles. Avec ces attentes, il est jugé inutile de normaliser cette interface.

NOTE Cette interface est une matière d'une potentielle normalisation future.

- Interfaces externes: Un important attribut de la présente norme est qu'elle est conçue pour permettre au réseau sans fil d'exercer un effet de levier sur une infrastructure de communications d'une installation ou de s'y intégrer. La présente norme définit les rôles spécifiques pour permettre à ce réseau de s'interfacer à d'autres réseaux, y compris les réseaux tant câblés que sans fil et les réseaux tant normalisés que propriétaires. Cependant, de même que ces réseaux externes spécifiques ne peuvent pas être identifiés par la présente norme, de même les interfaces à ces réseaux ne peuvent pas être identifiées ou spécifiées.

4.5.2 Structures des données

4.5.2.1 PDU définies

La présente norme définit la structure des unités de données de protocole (PDU) utilisées pour la communication entre appareils aux couches de protocoles suivantes: PhL, DL, NL, TL et AL. La plupart de celles-ci sont basées sur d'autres normes internationales: des normes ISO/IEC/IEEE pour la PhL et la DL, et des normes IETF pour la NL et la TL. Toutes ces définitions de PDU sont normatives, soumises à des essais externes d'examen et de conformité.

Du point de vue conceptuel, chaque classe distincte de PDU a:

- une syntaxe de transfert abstraite, qui décrit la structure de la PDU, y compris l'ordre des champs et la signification sémantique de chaque champ et de son contenu alternatif; et
- une ou plusieurs syntaxes de transfert concrètes, qui décrivent le codage au sein de la PDU pour chacun de ces champs et variantes de contenu.

La présente norme spécifie une syntaxe de transfert concrète *complète* ou *canonique* pour chaque PDU et, pour les DPDU, les NPDU et les TPDU, elle spécifie également une syntaxe de transfert concrète *compressée* qui réduit les exigences énergétiques pour l'émission et la réception des PDU, ainsi que le temps d'occupation de la voie sans fil lorsque la PDU est en cours d'émission.

NOTE 1 L'occupation réduite de la voie réduit l'interférence avec d'autres appareils et systèmes sans fil, tout en augmentant la probabilité de réussite de la réception de la PDU. Elle réduit également la puissance moyenne exigée pour le fonctionnement des appareils, ce qui revêt une importance particulière pour les appareils qui ne sont pas connectés à une source d'alimentation externe.

Pour les DPDU et les TPDU, une troisième syntaxe concrète (à savoir le codage) est utilisée dans le calcul d'un code d'intégrité de message pour la PDU. Le codage en question ajoute habituellement un *pseudo-en-tête* en préfixe à la PDU telle qu'émise/reçue, lequel pseudo-en-tête contient des informations spécifiques d'adressage de PDU de couche inférieure, servant ainsi à lier ces informations de couche inférieure à la PDU dont l'intégrité est couverte par le MIC calculé. Dans la présente norme, cette troisième syntaxe concrète est appelée la *syntaxe de calcul de MIC*.

NOTE 2 Les descriptions de PDU dans la présente norme réunissent souvent la syntaxe abstraite de la PDU (à savoir son contenu logique) avec la syntaxe de transfert concrète (à savoir son codage). Il est prévu qu'une édition future de la présente norme corrigera ce défaut.

4.5.2.2 Structures de données de gestion définies

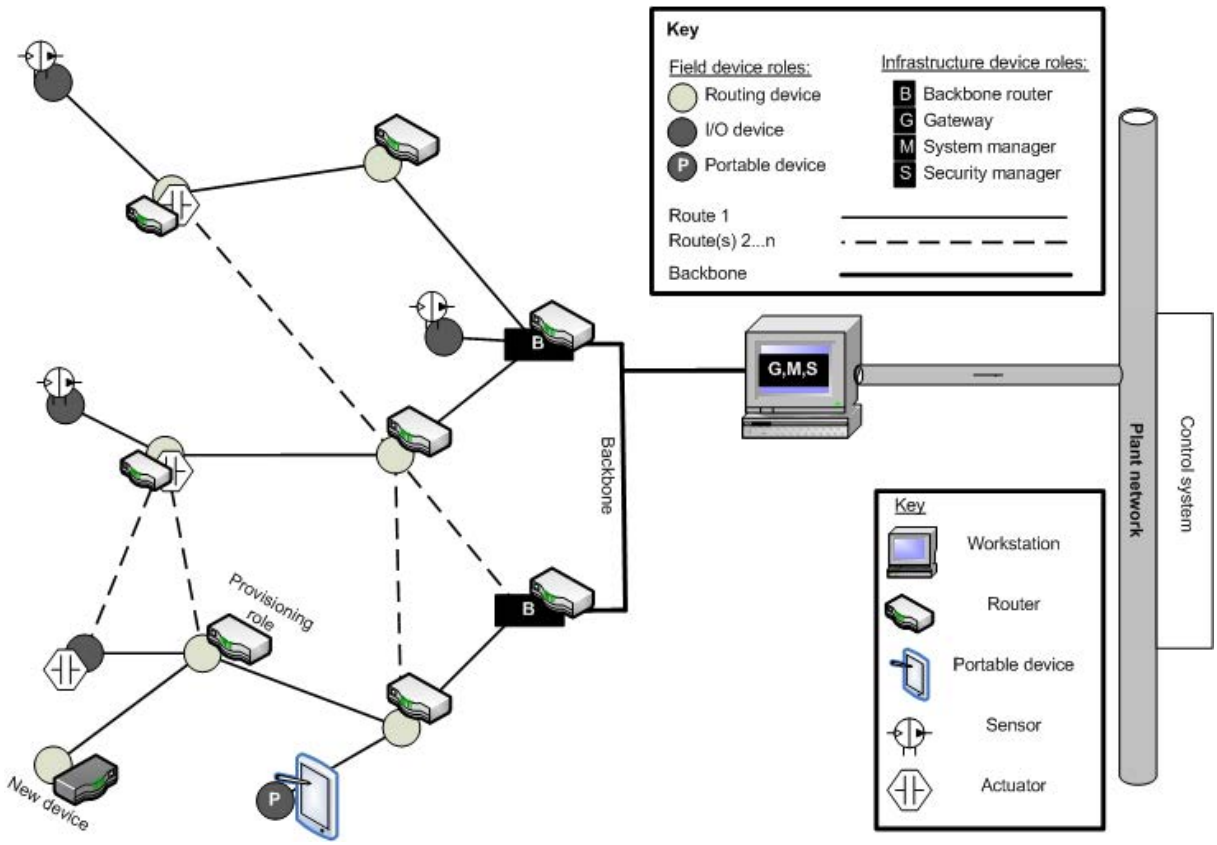
La présente norme définit la structure des objets de données de gestion aux diverses couches de protocoles. Ces définitions sont normatives quant au comportement des opérations définies sur ces structures de données et quant à la présentation de ces structures de données dans la mesure, et sous la forme, où elles apparaissent lorsqu'elles sont acheminées par des PDU. Cependant, la représentation de ces structures de données internes à une mise en œuvre ne s'inscrit pas dans le domaine d'application de la normalisation, car cette représentation n'est pas observable et n'est donc pas soumise à des essais de conformité.

4.5.3 Description de réseau

La Figure 1 montre les zones de communication adressées par la présente norme, ainsi que les zones (ombrées en bleu) qui ne s'inscrivent pas dans le domaine d'application de la présente norme. Dans la Figure 1, les objets circulaires représentent les rôles pour les appareils de terrain (capteurs, robinets, actionneurs, etc.) tandis que les objets rectangulaires représentent les rôles pour les appareils d'infrastructure qui communiquent avec d'autres appareils de réseau par une interface au réseau dorsal de l'infrastructure de réseau.

NOTE La présente norme définit les rôles dont les appareils sont les modes de réalisation; pour plus d'informations sur ces rôles, voir 5.2.6.

Une dorsale est un réseau de données (préférentiellement de haut débit de données) qui n'est pas défini par la présente norme. Cette dorsale peut être un réseau Ethernet industriel, IEEE 802.11, ou tout autre réseau au sein du moyen s'interfaçant au réseau de l'installation. Voir l'Annexe E pour plus d'informations et hypothèses relatives aux caractéristiques d'une dorsale.



Légende

Anglais	Français
New device	Nouvel appareil
Provisioning role	Rôle de configuration
Backbone	Dorsale
Key	Clé
Field device roles	Rôles d'appareil de terrain
Routing device	Appareil de routage
I/O device	Appareil E/S
Portable device	Appareil portatif
Infrastructure device roles	Rôles d'appareil d'infrastructure
B Backbone router	B Routeur dorsal
G Gateway	G Passerelle
M System manager	M Gestionnaire système
S Security manager	S Gestionnaire de sécurité
Route 1	Chemin 1
Route(s) 2…n	Chemin(s) 2…n
Backbone	Dorsale
Key	Clé
Workstation	Station de travail
Router	Routeur

Anglais	Français
Portable device	Appareil portatif
Sensor	Capteur
Actuator	Actionneur
Control system	Système de commande
Plant network	Réseau d'installation

Figure 1 – Réseau conforme à la norme

Un réseau complet tel que défini dans la présente norme comprend tous les composants et protocoles exigés pour acheminer un trafic sécurisé, gérer des ressources réseau et s'intégrer à des systèmes hôtes. Un réseau complet est constitué d'un ou plusieurs sous-réseaux D de terrain qui peuvent être connectés par un appareil d'infrastructure à un sous-réseau d'installation.

Un sous-réseau D de terrain est constitué d'un ensemble d'appareils de terrain qui communiquent sans fil en utilisant une pile de protocoles définie par la présente norme. Comme illustré à la Figure 1, certains appareils de terrain peuvent avoir des capacités d'acheminement, leur permettant de transmettre des messages issus d'autres appareils.

Un sous-réseau de transit est constitué d'appareils d'infrastructure sur une dorsale, tels que des routeurs dorsaux, des passerelles, des gestionnaires de système et des gestionnaires de sécurité. Comme le moyen de communication physique de la dorsale et sa pile de protocoles de réseau ne relèvent pas du domaine d'application de la présente norme, ils ne sont pas spécifiés et peuvent inclure la tunnellisation des PDU conformes sur des protocoles TL ou AL externes.

Les appareils qui connectent deux sous-réseaux D disparates ont au moins deux interfaces DLE et PhLE. Un routeur dorsal connecte un sous-réseau D de terrain à un sous-réseau dorsal D. Une passerelle connecte un sous-réseau dorsal à un sous-réseau d'installation; elle peut être co-allouée à un routeur dorsal.

NOTE 1 La portée d'un sous-réseau dépend de la couche de communication supérieure utilisée dans la construction du sous-réseau, qui pourrait être la couche OSI 1, 2, 3 ou 4.

NOTE 2 Etant donné que les passerelles ne sont pas définies dans la présente norme, la nature de l'interface qu'elles fournissent à un réseau "d'installation" est strictement théorique. Ainsi, le terme "réseau d'installation" (Figure 1) désigne un réseau ou tout autre moyen de communication résidant à l'extrémité de la passerelle par rapport aux sous-réseaux D qui sont couverts par la présente norme. De même, le terme "sous-réseau de terrain" désigne le sous-réseau D composé directement des appareils de terrain.

Tout l'adressage, le routage et le transport sont limités à la portée du sous-réseau D de terrain. Chaque DLE au sein d'un tel sous-réseau D est identifié par une adresse DL16Address locale, ainsi qu'une adresse IPv6Address avec une portée globale.

4.5.4 Construction d'une unité de données de protocole générique

La présente norme de communication utilise des couches de protocole de communication modélisées selon le Modèle de référence de base OSI. Une couche de protocole encapsule généralement les données qu'elle achemine pour une couche de protocole supérieure, puis utilise à son tour une couche inférieure pour acheminer le résultat encapsulé.

Les informations acheminées sur le réseau entre des entités homologues s'appellent une unité de données de protocole (PDU). Les informations acheminées à la frontière d'une couche entre les entités de couche d'un même nœud de réseau s'appellent une unité de données de service (SDU). Une SDU se compose généralement d'une seule PDU, mais elle peut également comprendre un groupe de PDU concaténées (voir "concaténation", 3.1.1.16).

Une SDU est considérée comme une chaîne d'octets ou de bits opaque devant être acheminée de manière transparente (autrement dit sans interprétabilité ni altération) par la

couche inférieure. La SDU peut être acheminée par une unité PDU de couche inférieure, ou être segmentée (voir "segmentation", 3.1.1.58) afin d'être acheminée de manière individuelle par plusieurs PDU de couche inférieure, ou être regroupée avec d'autres SDU (voir "groupage", 3.1.1.10) afin d'être acheminée de manière groupée dans une seule PDU de couche inférieure. Dans le cas le plus courant, un en-tête et un bas de page sont ajoutés à une SDU unique pour former une PDU unique (voir Figure 2).

NOTE Le bas de page est habituellement vide ou utilisé pour une certaine forme de code d'intégrité de message (MIC), telle qu'une somme de contrôle ou une partie de code de hachage à clé sécurisé de manière cryptographique simple facile à reproduire.

L'en-tête et le bas de page sont souvent appelés surdébit (par rapport à la SDU ou aux SDU acheminées sous forme individuelle, groupée ou segmentée), la quantité de surdébit dépendant de la quantité des informations complémentaires ayant besoin d'être incluses pour que le protocole d'acheminement fonctionne correctement. Un but de la présente norme étant de réduire au maximum la consommation d'énergie et l'occupation de voie lors de l'acheminement des PDU, la réduction au maximum de la quantité de surdébit à chaque couche de protocoles est donc une méthode principale d'atteindre le but en question.

Une description complète de chaque en-tête et bas peut être consultée dans la description appropriée de la couche de protocoles. Une PDU complète à plusieurs couches comprend tous les en-têtes et tous les bas de page, tels que montrés à la Figure 3. La quantité de données (mesurée en octets) d'une PDU d'application qui peut être envoyée en une seule émission est déterminée par la différence entre la charge utile de PhL maximale autorisée ou prise en charge et le surdébit imposé par tous les en-têtes et de tous les bas de page intermédiaires.

Légende

Anglais	Français
Header	En-tête
SDU	SDU
Footer	Bas de page

Figure 2 – PDU typique à une seule couche sans fragmentation ni groupage

Légende

Anglais	Français
PhL header	En-tête PhL
DL header	En-tête DL
NL header	En-tête NL
TL header	En-tête TL
Application PDU	PDU d'application
TL footer	Bas de page de TL
NL footer	Bas de page de NL
DL footer	Bas de page de DL

Figure 3 – Structure de PDU complète à plusieurs couches utilisée par la présente norme

4.5.5 Données abstraites et représentations concrètes

4.5.5.1 Types de données abstraites

Chaque couche de protocoles de la présente norme définit la structure des PDU qu'elle échange avec des entités de protocole homologues sur la même couche, ainsi que des SDU et des informations de contrôle d'interface associées (ICI) qu'elle échange avec des entités de couche adjacentes au sein d'un nœud local du réseau.

Par ailleurs, chaque couche de protocoles définit les structures de données de gestion qui sont échangées par chaque entité de couche (sous la forme de données acheminées) avec des entités de gestion de systèmes distants.

Chacune de ces structures de données possède une forme abstraite qui exige une représentation concrète. Les éléments abstraits sont soit des valeurs scalaires, soit des valeurs composites composées de valeurs scalaires et d'autres valeurs composites. Les valeurs composites peuvent être homogènes, auquel cas il s'agit de groupes monodimensionnels ou multidimensionnels (souvent connues sous le nom de "vecteur" ou "matrice", respectivement), ou bien hétérogènes (souvent appelées simplement "structure de données").

Les éléments scalaires utilisés par les entités de protocole de la présente norme sont:

a) des entiers d'une plage restreinte, où chaque valeur abstraite est représentée par la valeur binaire concrète à deux compléments équivalente ou une valeur binaire concrète non signée, selon que la plage abstraite inclut des valeurs négatives;

b) des énumérations, généralement déclarées sous la forme d'une représentation UnsignedN pour $N \leq 8$, où chaque valeur abstraite est généralement représentée par l'index ordinal d'origine zéro de la valeur abstraite parmi la liste de valeurs définies;

c) des booléens, possédant deux valeurs abstraites (false, true) auxquelles les règles et les opérateurs de la logique booléenne s'appliquent, ainsi qu'une représentation concrète de false par zéro et de true par une valeur binaire différente de zéro;

NOTE 1 Les booléens tirent leur nom du logicien George Boole.

NOTE 2 Même si les booléens semblent être une classe d'énumération spéciale, les différences sont que la valeur true peut être représentée par une valeur binaire différente de zéro et que les opérateurs booléens s'appliquent à cette classe.

d) des nombres à virgule flottante IEEE d'une plage et d'une précision spécifiques, dont les valeurs sont des nombres réels approximatifs ou des constantes non-numériques spéciales;

e) des représentations du Temps atomique international (TAI) correspondant à des valeurs entières ou à des valeurs à virgule fixe à l'échelle modulo 2^{32} s.

Les éléments composites utilisés par les entités de protocole de la présente norme incluent trois classes de groupes d'origine zéro monodimensionnels, appelés chaînes (*strings*). Ce sont:

f) les caractères, appelés chaînes visibles;

g) les bits, appelés chaînes de bits; et

h) les octets non interprétés, appelés chaînes d'octets.

Les autres éléments composites incluent les groupes booléens condensés, qui sont souvent (à tort) assimilés à leur représentation sous-jacente en tant que chaînes de bits.

4.5.5.2 Déclarations des éléments de données abstraites et de leurs représentations concrètes

4.5.5.2.1 Déclarations simples

Dans le cadre de la présente norme, un type abstrait et sa représentation concrète sont souvent déclarés sous une forme unifiée qui indique à la fois la classe de 4.5.5.1 a) à h) ainsi que le nombre de bits dans la représentation binaire sous-jacente.

- Les entiers sont déclarés au sein d'une plage implicite sous la forme *unsignedN* (impliquant une plage de $0..2^N-1$) ou *signedN* (impliquant une plage de $-2^{N-1}..2^{N-1}-1$), où N représente le nombre de bits de la représentation, généralement un multiple de 8.

  Les entiers qui exigent une plage différente de celle impliquée par leur représentation sont déclarés sous la forme

  unsignedN plage min..max

  où min et max représentent respectivement les valeurs minimale et maximale de la plage de cet élément entier.

- Les booléens sont déclarés sous la forme *BooleanN*, où N représente le nombre de bits de la représentation, généralement 1 (lorsqu'ils se trouvent dans une structure de données condensée) ou 8.

- Les nombres à virgule flottante, ainsi que leurs plage et précision sont déclarés sous la forme *float32* ou *float64*.

- Les chaînes de taille fixe ainsi que leurs tailles sont déclarées sous les formes *visibleStringN*, *octetStringN* et *bitStringN*, où N représente le nombre d'éléments dans le groupe sous-jacent.

NOTE 1 Un mécanisme de codage interne dans certaines chaînes visibles, généralement une valeur nulle (0x00) utilisée en guise de délimiteur de fin de contenu, est souvent utilisé pour tronquer la taille effective de la chaîne contenue. De nombreuses bibliothèques d'opérateurs de chaîne présument un tel codage.

- Les chaînes de taille variable sont déclarées sous les formes *visibleString* et *octetString* sans taille déclarée concaténée (autrement dit la valeur N de *visibleStringN*).

- Les groupes booléens condensés ainsi que leur taille sont déclarés sous la forme *BooleanArrayN*, où N représente le nombre d'éléments dans le groupe (et par conséquent le nombre de bits de la représentation).

- Les constantes nommées, généralement issues de valeurs de champs UnsignedN, peuvent être déclarées à la manière de l'ASN.1 comme s'il s'agissait d'énumérations ASN.1, comme dans:

  UnsignedN {name-1(integer-value-1), … , name- K(integer-value-K)}

  où N représente le nombre de bits de la représentation et où une déclaration explicite des constantes est fournie sous la forme d'une liste séparée par des virgules et placée entre crochets, chaque élément étant suivi de "(K)" où K représente la valeur affectée à cet élément.

  EXEMPLE 1 Une déclaration pour les couches de protocoles définies dans la présente norme pourrait être:

  protocolLayers Unsigned3 {PhL(1), DL(2), NL(3), TL(4), AL(7)}

  NOTE 2 L'exemple 1 démontre que les noms des éléments de ce qui équivaut à une énumération peuvent ne pas être disjoints de ceux utilisés en d'autres endroits de la présente norme, comme cela pourrait être exigé dans le cadre d'une spécification de langage de programmation explicite.

  Sinon, ces constantes nommées peuvent être déclarées sous la forme

  UnsignedN {integer-value-1:description-1; … ; integer-value-K:description-K}

  où N représente le nombre de bits de la représentation et où une déclaration explicite des constantes nommées est fournie sous la forme d'une liste séparée par des points-

virgules et placée entre crochets, chaque élément étant précédé de "K" où KK représente la valeur affectée à cette constante nommée. (Voir 3.3.5.)

EXEMPLE 2 A Une déclaration pour la méthode de jointure join_method définie dans la présente norme pourrait être:

join_method Unsigned8{ 0:none; 1:join and start; 2:warm restart; 3:restart as provisioned; 4:reset to factory defaults }

Pour cette dernière forme de déclaration, la liste entre crochets peut être séparée de la déclaration de la représentation du champ.

EXEMPLE 3 join_method Unsigned8

Constantes nommées:
0: none
1: join and start
2: warm restart
3: restart as provisioned
4: reset to factory defaults

NOTE 3 On retrouve souvent cette dernière forme de déclaration dans les descriptions tabulaires de structures de données dans la présente norme où la première partie de la déclaration figure dans une colonne et la deuxième partie figure dans la même colonne ou dans une colonne différente de la même ligne.

Souvent, seuls quelques éléments de la plage d'une déclaration UnsignedN sont nommés, un tel cas pouvant survenir lorsque la valeur tout-à-zéro ou tout-à-un de la représentation a une interprétation spéciale. A certaines occasions, en particulier lorsque la description est "réservée", une plage de valeurs peut être spécifiée à la place d'une valeur unique.

4.5.5.2.2 Déclarations d'objets composés et de méthodes avec leurs arguments

Dans la présente norme, les structures de données composées sont généralement déclarées sous la forme de tables, chaque ligne (après les lignes d'en-tête) décrivant un élément constituant au sein de la structure de données.

Dans la présente norme, les descriptions de méthodes sont généralement déclarées sous la forme de tables. Chacune de ces descriptions spécifie un nom de méthode, un ID de méthode numérique ainsi qu'une description de la méthode, suivis d'une série de descriptions d'arguments d'entrée de méthode, suivis d'une série de descriptions d'arguments de sortie de méthode. Comme pour les définitions de structures de données, chaque argument est déclaré tout seul sur une ligne de la table et possède un type déclaré ainsi que, lorsque cela s'avère approprié, une déclaration des variantes pour les constantes nommées associées.

4.6 Caractéristiques de réseau

4.6.1 Généralités

Les caractéristiques d'un WISN (à savoir un réseau sans fil qui est conforme à la présente norme) sont notamment:

- évolutif;
- extensible;
- prise en charge d'un fonctionnement simple;
- exploitation sans licence;
- robustesse en présence d'interférence et avec des réseaux autres que WISN;
- déterminisme ou accès aux supports sans conflits;
- réseau auto-organisé avec prise en charge des communications redondantes de l'appareil de terrain vers le réseau d'installation;

NOTE La prise en charge de la redondance n'est pas définie dans la présente norme.

- couche réseau compatible avec IP;
- coexistence avec d'autres appareils sans fil dans l'espace de travail industriel;
- sécurité, y compris l'authenticité des données, la confidentialité des données, l'intégrité des données, la protection contre les retards et la protection contre le rejet;
- gestion de système de tous les appareils de communication;
- prise en charge des processus applicatifs utilisant des objets normalisés; et
- prise en charge de la tunnellisation (à savoir le transport) d'autres protocoles à travers le réseau sans fil.

4.6.2 Evolutivité

L'architecture prend en charge des systèmes sans fil qui s'étendent dans la plage physique allant d'un seul petit sous-réseau D isolé, tel qu'on pourrait en trouver au voisinage d'un puits de gaz ou de pétrole ou un très petit atelier d'usinage, aux systèmes intégrés de plusieurs milliers d'appareils et plusieurs sous-réseaux D qui peuvent couvrir une installation de plusieurs kilomètres carrés. Il n'y a aucune limite technique au nombre d'appareils pouvant participer à un réseau composé de plusieurs sous-réseaux D. Un sous-réseau D, à savoir un groupe de DLE partageant un certain nombre d'aspects de configuration de DL, peut contenir jusqu'à 30 000 DLE (qui est une limitation de l'espace d'adressage d'un sous-réseau D). Avec plusieurs sous-réseaux D, le nombre des DLE (et donc des appareils) dans le réseau peut évoluer linéairement.

La quantité maximale de données de couche supérieure ou de gestion pouvant être acheminées dans une seule DPDU est limitée par la PhL et la quantité de surdébit de DL exigée. Par conséquent, la présente norme prend en charge la fragmentation au sein de la DL, permettant l'émission d'une quantité de données bien plus élevée. Dans la fragmentation, les données sont segmentées en portions de dimension appropriée au niveau de la DLE émettrice, encapsulées dans des DPDU et émises à travers le sous-réseau D pour être ensuite réassemblées au niveau de la DLE destinataire. Une utilisation de ce mécanisme est la mise à jour du firmware d'appareil.

4.6.3 Extensibilité

Les protocoles définis par la présente norme ont des champs et des plages de valeurs de paramètres qui sont réservés pour usage futur et des identificateurs de version (édition) dans des en-têtes qui permettent l'identification de l'édition appropriée.

NOTE Ces fonctionnalités sont destinées à permettre de futures révisions de la présente norme pour offrir une fonctionnalité complémentaire ou améliorée sans sacrifier inutilement la compatibilité amont, ni alourdir le codage (ce qui survient généralement lors de l'utilisation du mécanisme d'extensibilité déclaré ASN.1).

4.6.4 Fonctionnement simple

A la suite de la configuration, telle que décrite à l'Article 13, une DLE peut automatiquement se rattacher au sous-réseau D et à son réseau N supérieur. Le rattachement automatique d'appareil et la formation d'un sous-réseau D permettent la configuration du système avec un besoin minimal en personnel ayant la formation spécialisée en ondes radioélectriques (RF) et en outils.

En outre, la présente norme prend en charge l'utilisation de techniques de routage D totalement redondantes et autorégénératrices pour réduire au maximum la maintenance du sous-réseau D. Voir 9.1.6 pour plus d'informations.

4.6.5 Exploitation sans licence sur site

La présente norme utilise des radios conformes à l'IEEE 802.15.4 utilisant les voies DSSS 11..26 de 2,4 GHz spécifiées dans la norme en question.

NOTE 1 La bande ISM de 2,4 GHz est disponible et exempte de licence sur site dans la plupart des pays, sous réserve que l'équipement dispose d'une licence de type pour une telle exploitation acceptée dans le pays en question.

NOTE 2 L'ISA TR100.00.01 fournit des informations complémentaires sur l'exploitation des ondes radioélectriques.

4.6.6 Robustesse en présence d'interférence, y compris en provenant d'autres systèmes sans fil

La présente norme utilise le saut de voies en fonction du temps

- pour assurer un niveau d'immunité contre les interférences issues d'autres appareils RF fonctionnant dans la même bande,
- pour atténuer les effets des interférences à chemins multiples,
- pour faciliter la coexistence avec d'autres systèmes RF, et
- pour satisfaire à des exigences réglementaires communes.

Dans certains régimes de réglementation, la coexistence peut être encore plus renforcée par une mise sélective en liste noire de voies, évitant ainsi des voies autrement occupées dans la bande. La mise sélective en liste noire de voies peut aussi renforcer la fiabilité en évitant l'utilisation des voies avec une performance constamment médiocre.

4.6.7 Déterminisme et accès sans conflits aux supports

La présente norme définit un mécanisme d'accès multiple par répartition dans le temps (AMRT) qui permet à un appareil d'accéder au support RF sur un programme et, de ce fait, la majeure partie de la concurrence pour l'utilisation de la voie aura été préalablement résolue par l'agent de programmation. Les communications synchronisées sont fondées sur des intervalles de temps consécutifs qui ont des durées configurables, généralement dans la plage de 10 ms à 12 ms. La programmation et le fonctionnement des DLE sont grandement simplifiés lorsque tous les intervalles de temps ont une seule et même durée et, donc, les WISN sont en général configurés pour n'avoir qu'une seule durée d'intervalle de temps utilisée pour tous les intervalles de temps.

NOTE 1 Etant donné que les intervalles de temps sont attribués à des voies logiques qui sont ensuite associées de manière cyclique à des voies physiques, l'utilisation d'une seule durée d'intervalle de temps commune signifie que l'évitement de contention sur les voies logiques élimine automatiquement cette contention sur les voies physiques. Des intervalles de temps de durées différentes font perdre cette simplification de planification.

Une DLE expéditrice reçoit un intervalle de temps et une voie qui lui sont assignés et qui sont propres à l'appareil en question et à l'appareil avec lequel elle communique. Ces durées d'intervalle de temps sont configurables sur une base supertrame par supertrame. Une supertrame est un ensemble cyclique d'intervalles de temps. L'aptitude de configurer la durée d'intervalle de temps permet

- de plus courts intervalles de temps pour tirer pleinement profit de mises en œuvre optimisées;
- de plus longs intervalles de temps pour prendre en charge
  - des durées prolongées d'attente de DPDU,
  - un acquittement sériel de deux ou plusieurs appareils configurés (par exemple, duodiffusion), et
  - un accès CSMA/CA au début d'un intervalle de temps (par exemple, pour mettre en œuvre le protocole Listen Before Talk ou pour bénéficier d'un accès priorisé à des intervalles de temps partagés); et
- des périodes de durée prolongée pour les sauts de voie lents.

NOTE 2 La réglementation locale peut contraindre la durée maximale d'une telle période de saut de voie lent.

La prise en charge est assurée aussi bien pour les intervalles de temps dédiés au trafic prévisible et régulier que pour les intervalles de temps partagés pour le trafic par salves tel que les alarmes. L'édition/abonnement, les communications client/serveur, les rapports d'alerte et le transfert en masse de données sont également pris en charge.

4.6.8 Mise en réseau auto-organisé avec prise en charge de redondance

Les techniques de routage pleinement redondantes et autorégénératrices, telles que le routage maillé (voir 9.1.6), prennent en charge la fiabilité de réseau de bout en bout face à des conditions changeantes d'ondes RF et d'environnement. Les caractéristiques spéciales qui permettent au réseau d'adapter les fréquences utilisées (saut de voie adaptatif, par exemple) avec le routage maillé peuvent automatiquement atténuer les problèmes de coexistence sans intervention de l'utilisateur.

4.6.9 NL compatible avec le protocole IP

La NL de la présente norme utilise des formats d'en-tête qui se conforment aux normes 6LoWPAN de l'Internet Engineering Task Force, facilitant ainsi l'utilisation potentielle des réseaux compatibles à l'IPv6 comme réseaux dorsaux en appui à la présente norme. L'utilisation d'en-têtes qui se conforment à la 6LoWPAN n'implique ni

- qu'un réseau dorsal a besoin être basé sur la 6LoWPAN ou l'IPv6, ni
- qu'un réseau basé sur la présente norme est ouvert au piratage par Internet.

En fait, un grand nombre des réseaux basés sur la présente norme ne sont pas directement connectés à Internet. Les autres peuvent utiliser la vielle norme IPv4, au moins pendant les premières années d'exploitation.

NOTE L'utilisation d'IPv6 permet l'utilisation de logiciels et d'outils réseau normalisés, ainsi que la possibilité d'utiliser une large gamme de normes IETF dans les révisions ou extensions futures de la présente norme.

4.6.10 Coexistence avec d'autres systèmes de fréquences radioélectriques

4.6.10.1 Vue d'ensemble de la coexistence

L'architecture de système spécifiée par la présente norme est spécifiquement conçue pour prendre en charge la coexistence avec d'autres

- WISN (à savoir des systèmes sans fil conformes à la présente norme);
- d'autres réseaux de communication opérant à 2,4 GHz qui utilisent différentes versions des normes IEEE 802.11, IEEE 802.15.1 et IEEE 802.15.4; et
- d'autres appareils qui utilisent le même spectre de fréquences radioélectriques.

Le fonctionnement avec de très courtes communications synchronisées tend à réduire l'encombrement des bandes RF et à permettre à des systèmes voisins de récupérer rapidement de leurs PhPDU perdues ou corrompues.

En raison du temps de séjour réduit sur une voie quelconque au moment d'un saut de voie, l'impact sur les autres systèmes radioélectriques est réduit et la fiabilité face à des interférences est accrue. Par exemple, les DPDU peuvent être renvoyées sur d'autres voies non perturbées. La mise sélective en liste noire de voies augmente encore plus la coexistence en évitant les voies qui sont prédéterminées comme étant inutilisables ou trop encombrées.

La présente norme prend en charge (mais n'exige pas) l'utilisation de l'évaluation de voie libre (CCA) pour réduire au maximum les collisions avec des systèmes non synchronisés et aussi pour fournir la fonctionnalité CSMA/CA au sein de systèmes synchronisés.

NOTE Certaines juridictions réglementaires exigent l'utilisation de la CCA dans certains modes d'exploitation. Une telle utilisation exigée est fournie par la présente norme lorsqu'un appareil est configuré dans le but d'être exploité dans ces juridictions.

L'architecture des WISN est conçue pour prendre en charge le fonctionnement en présence d'interférences issues de radiateurs intempestifs, tels que les fours micro-ondes, en utilisant le saut de voie et un protocole de demande automatique de répétition (ARQ. L'ARQ est une méthode courante de contrôle d'erreurs pour l'émission de données qui utilise des acquittements pour la réception réussie des messages, couplée avec la réémission différée en cas de réception erronée, pour assurer l'acheminement fiable des données.

Pour des informations complémentaires relatives à la diversité des techniques qui maximalisent la coexistence, voir 9.1.2.

4.6.10.2 Stratégies de coexistence

4.6.10.2.1 Généralités

Ci-après sont donnés des exemples de techniques de coexistence qui ne sont spécifiques à aucun protocole. Ils améliorent la coexistence avec une large gamme d'appareils partageant la bande de 2,4 GHz tout en optimisant le succès de chaque tentative de communication.

NOTE Voir l'IEC 62657-2 pour un débat plus complet sur la coexistence sans fil.

4.6.10.2.2 Infrastructure à effet de levier pour les liaisons de communications à haut débit de données

Les réseaux à sauts multiples acheminent plusieurs fois, une fois (au moins) par saut, les mêmes données de couche supérieure. Une capacité fondamentale de la présente norme est l'aptitude à acheminer aussi directement que possible les données à une DLE connectée à un sous-réseau dorsal (préférentiellement un réunissant un haut débit de données et un faible taux d'erreurs). Cela réduit souvent l'utilisation de la PhL spécifiée par la présente norme à un(e) ou deux transactions D et sauts de sous-réseau D pour chaque PDU de couche supérieure acheminée.

4.6.10.2.3 Fonctionnement par intervalles de temps

Le fonctionnement par intervalles de temps et les émissions programmées servent à réduire au maximum les collisions au sein du sous-réseau D et, de ce fait, à éviter l'inutile utilisation de la voie pour les répétitions de tentative.

4.6.10.2.4 Sélection du type d'ondes radioélectriques

Le sous-ensemble 2,4 GHz de l'IEEE 802.15.4 a été sélectionnée comme PhL pour la présente norme, car, dans un grand nombre de conditions, des ondes radioélectriques semblables se chevauchant, ainsi que les ondes radioélectriques de l'IEEE 802.11, peuvent être actives simultanément sans perte des données acheminées.

NOTE La présente norme s'intéresse essentiellement à la coexistence avec les versions plus récentes de ces normes, alors qu'on tend à rencontrer les anciennes versions de moins en moins souvent avec le temps

4.6.10.2.5 Faible facteur d'utilisation

L'acheminement de données pour les applications focus décrites en 0.1 est rare, alors que le surdébit ajouté issu des couches de protocoles d'envoi est réduit au maximum.

4.6.10.2.6 Emissions saccadées

Les émissions attendues sont très courtes, ce qui est une caractéristique de la PhL sélectionnée. Cela permet à des réseaux IEEE 802.11 co-implantés de récupérer rapidement en cas d'interférences provenant du WISN.

4.6.10.2.7 Diversité temporelle

Plusieurs des applications focus ont des exigences relatives à la latence moins strictes que les autres utilisateurs du spectre, fournissant plus d'opportunité d'utiliser la diversité temporelle pour la coexistence. Des périodes configurables de répétitions de tentatives, s'étendant potentiellement sur des centaines de millisecondes, permettent au système de coexister avec d'autres utilisateurs qui peuvent exiger l'utilisation du même spectre au cours des salves d'activité de plus haute priorité.

4.6.10.2.8 Diversité de voie

Le faible facteur d'utilisation des ondes radioélectriques est étalé sur seize voies IEEE 802.15.4, ce qui réduit encore plus le potentiel de cas le plus défavorable pour les interférences à 1 % du temps voire moins dans un grand nombre de scénarios réalistes.

4.6.10.2.9 Gestion de spectre

L'utilisateur peut configurer des supertrames dans le sous-réseau D afin de limiter le fonctionnement de certaines voies radioélectriques.

4.6.10.2.10 Utilisation sélective de voie

Lorsque le régime de réglementation le permet, la gestion D peut éviter les voies problématiques sur une base liaison par liaison, telles que les voies qui présentent une interférence croisée IEEE 802.11 ou des évanouissements persistants sur des trajets multiples.

4.6.10.2.11 Evitement des collisions

Toutes les DLE prennent en charge l'accès CSMA/CA, qui permet à une DLE de mettre en œuvre le protocole Listen Before Talk pour permettre la détection en temps réel de l'utilisation de la voie et retarder sa propre émission, réduisant les interférences à ces autres utilisateurs.

NOTE Un tel évitement est exigé dans certains régimes de réglementation, du moins dans certains modes d'exploitation.

4.6.10.2.12 PhPDU variables

En raison des mesures de sécurité intégrées de la DL, qui incluent la défense contre des attaques de rejet dans la même voie et à travers les voies, les PhPDU varient d'une émission à l'autre même pendant la réémission des mêmes informations de DPDU nominales. Avec la modulation à étalement de spectre, cela engendre des interférences de durée variable même lorsque des messages sinon identiques sont en cours de transmission.

4.6.11 Transactions D par intervalles de temps à voie assignée comme base pour la communication

4.6.11.1 Vue d'ensemble

Excepté pendant l'intervalle où une DLE sollicite l'occasion de rejoindre un sous-réseau D, chaque instance de la communication de DLE selon la présente norme se produit

a) dans une fenêtre temporelle spécifiée au préalable, par rapport au sens du temps TAI de la DLE;

b) sur une voie spécifique de PhL, à un niveau de puissance qui respecte les réglementations locales;

c) en utilisant un modèle d'intervalle de temps spécifique pour l'initiateur d'une transaction D, qui spécifie

   1) l'acquisition de voie, configurée en accord avec les réglementations locales et le modèle d'intervalle de temps;

2) l'émission d'une DPDU Data (à savoir la DPDU initiale d'une transaction) contenant soit des données de couche supérieure, soit des données de gestion, à destination d'un ensemble de correspondants prévus; et

3) lorsque cela est ainsi spécifié par le modèle d'intervalle de temps, la réception tentée d'une ou plusieurs DPDU ACK/NAK (à savoir le message SCS) envoyées par des correspondants prévus;

NOTE 1 La réception intentionnelle de plus d'une DPDU ACK/NAK est utile pour évaluer le fonctionnement d'un réseau, mais n'est pas essentielle pour l'acheminement réussi d'une DPDU.

d) en utilisant un différent modèle d'intervalle de temps spécifique pour un correspondant prévu d'une transaction D, qui spécifie

1) la durée de la phase d'acquisition de voie, relative à c)1);

2) la réception tentée d'une DPDU Data contenant soit des données de couche supérieure, soit des données de gestion, adressées soit à la DLE elle-même, soit à une autre adresse DL16Address spécifiée; et

NOTE 2 Cette dernière capacité est utilisée pour la multidiffusion/diffusion et la duodiffusion/N-diffusion.

3) lorsque la réception d)2) s'est effectivement produite et était exempte d'erreur à la PhL avec une FCS de DLE sans erreur, et lorsque le modèle d'intervalle de temps le spécifie ainsi, l'émission d'une seule DPDU ACK/NAK (à savoir le message SCS) envoyée à l'adresse DL16Address d'expédition de la DPDU reçue en d)2), se produisant soit

i) à un retard spécifié après la fin de la réception de la DPDU Data d)2), soit

ii) à un instant spécifié avant la fin programmée de l'intervalle de temps,

comme le spécifie le modèle d'intervalle de temps.

Lorsque le modèle d'intervalle de temps correspondant pour l'initiateur de transaction spécifie plus d'un intervalle pour la réception de DPDU ACK/NAK en c)3), les modèles d'intervalle de temps pour les répondeurs de la transaction diffèrent en leurs valeurs assignées pour d)3)i) ou d)3)ii), assignant de ce fait ces réponses potentielles pour disjoindre les durées dans l'intervalle de temps.

Les appareils auxquels s'applique c)2) sont appelés des *initiateurs de transaction*; les appareils auxquels s'applique d)2) sont appelés des *destinataires de transaction* (qui constituent les destinataires prévus, et pas seulement des intermédiaires); les appareils auxquels s'applique d)3) sont appelés préférentiellement des *répondeurs de transaction* (même s'il s'agit également de *destinataires de transaction*).

NOTE 3 L'IEEE 802.15.4e spécifie des mécanismes qui sont similaires, mais pas identiques, à plusieurs des mécanismes DL spécifiés dans la présente norme.

4.6.11.2 Phase d'acquisition de voie

La phase d'acquisition de voie d'une transaction a deux utilisations.

a) **Listen Before Talk (LBT)**: Un mode de fonctionnement qui est exigé dans certaines juridictions de réglementation et facultatif dans d'autres, dont le but est de réduire l'interférence avec d'autres appareils émettant dans la même gamme de fréquences, que ces appareils utilisent des PhL semblables (d'autres systèmes IEEE 802.15.4 sur la même voie, par exemple) ou des PhL différentes qui se chevauchent dans leur utilisation de la fréquence (IEEE 802.11, par exemple,). Dans ce mode de fonctionnement, les initiateurs de transaction prévus échantillonnent la voie d'une manière spécifiée (par exemple, mode 1 de CCA) pendant une durée spécifiée, conformément aux exigences réglementaires locales, et arrêtent l'utilisation prévue de l'intervalle de temps si cet échantillonnage implique que le canal soit en cours d'utilisation par un autre courant appareil.

b) **CSMA/CA**: Un moyen par lequel plusieurs DLE conformes à la présente norme tentent de réclamer l'utilisation d'un intervalle de temps d'utilisation partagée. Dans ce mode de fonctionnement, chaque initiateur de transaction compétiteur fonctionne comme en a) pendant une durée qui est uniformément choisie dans une distribution de durées

d'intervalle qui augmente exponentiellement avec les échecs successifs (jusqu'à une certaine limite prédéterminée), fournissant de ce fait le repli exponentiel en cas d'encombrement pour l'utilisation de l'intervalle de temps. Quand la plus petite valeur de l'intervalle sélectionné est supérieure à zéro, un tel fonctionnement prend en charge l'accès priorisé, car les DLE qui mettent en œuvre le mode CSMA/CA tendent à différer à celles qui ne le font pas.

Les deux modes a) et b) peuvent être combinés pour satisfaire aux deux ensembles d'objectifs. Les modèles d'intervalle de temps utilisés par les destinataires prévus ont besoin de rendre compte de ces retards initiaux, comme en 4.6.11.1, d)1), afin que de tels destinataires ne mettent pas prématurément fin à la réception dans une tentative de réduire au maximum l'énergie utilisée par leurs récepteurs de PhL lorsqu'ils sont actifs.

En raison des propriétés bien connues de la propagation RF, les processus Listen Before Talk et CSMA/CA ci-dessus ne sont pas fiables quant à leur aptitude à détecter soit l'utilisation de voie par d'autres appareils, soit le potentiel que l'utilisation de voie pour une transaction donnée interfère avec d'autres communications distantes en cours. Cette question porte de nombreux noms, notamment celui fréquent de problème de "nœud caché". Ainsi le différé des transactions en raison de a) et/ou b) est toujours pessimiste en ce qui concerne l'interférence projetée, pourtant le non-différé est inadéquat pour éviter l'interférence (qui se produit aux récepteurs) causée par des émetteurs RF concourants qui échouent à se détecter les uns les autres.

4.6.11.3 Phase de communication

La phase de communication d'une transaction est utilisée pour trois classes de base de transactions, à savoir:

a) **Multicast/broadcast** (multidiffusion/diffusion): Le modèle d'intervalle de temps pour l'initiateur de transaction est constitué de 4.6.11.1, c)1) et c)2), avec omission de c)3), alors que le modèle d'intervalle de temps pour les destinataires de transaction est constitué de 4.6.11.1, d)1) et d)2), avec omission de d)3). Pour ces transactions, il n'y a pas de répondeur de transaction, car aucune occasion de réponse immédiate de message SCS n'est fournie dans le modèle.

b) **Unicast** (monodiffusion): Le modèle d'intervalle de temps pour l'initiateur de transaction est constitué de toutes les parties de 4.6.11.1, c), alors que le modèle pour les destinataires est constitué de toutes les parties de 4.6.11.1, d). Pour ces transactions, il y a exactement un répondeur de transaction prévu, avec une seule occasion de réponse immédiate ACK/NAK fournie dans le modèle.

c) **Duocast/N-cast** (duodiffusion/N-diffusion): Le modèle d'intervalle de temps pour l'initiateur de transaction est constitué de toutes les parties de 4.6.11.1, c), alors que le modèle pour les destinataires est constitué de toutes les parties de 4.6.11.1, d). Pour ces transactions, il y a un nombre désigné (respectivement 2 ou n pour la duodiffusion et la N-diffusion) de répondeurs de transaction prévus, chacun avec un modèle d'intervalle de temps différent qui spécifie une seule occasion de réponse pour une DPDU ACK/NAK disjointe de toutes les autres occasions de réponse pour la même transaction.

   Dans cette classe de transaction, tous les répondeurs de transaction ou tous les répondeurs (sauf le premier) se sensibilisent eux-mêmes pour recevoir une DPDU dont l'adresse DL16Address de destination n'est pas la propre adresse DL16Address de la DLE. Dans de tels cas, le modèle d'intervalle de temps spécifie également l'adresse DL16Address devant être utilisée à cet effet. Cette utilisation d'une adresse DL16Address distincte s'applique seulement aux DPDU Data de la transaction; toute DPDU ACK/NAK utilise les adresses DL16Address réelles explicites ou implicites du répondeur et de l'initiateur de la transaction.

4.6.12 Sécurité robuste et flexible

Tous les réseaux conformes ont un gestionnaire de sécurité pour gérer et authentifier des clés cryptographiques en transit. Des primitives de sécurité définies par l'IEEE 802.15.4 sont

utilisées par la DLE et la TLE, fournissant l'authentification de l'émetteur du message, l'intégrité du message, et la confidentialité facultative du contenu du message.

L'authentification de l'appareil est activée par l'utilisation de clés symétriques et d'identificateurs (ID) uniques d'appareil, avec une option pour l'utilisation de clés asymétriques pendant le processus de configuration de l'appareil et un certain nombre d'autres processus relatifs à la sécurité.

Au cours du fonctionnement normal, l'authenticité des données reçues et l'intégrité des données sont vérifiables par le biais de l'utilisation des clés symétriques secrètes connues à la fois de l'émetteur et du ou des destinataires.

Pendant la configuration, l'authenticité des authentifiants d'appareil reçus provenant d'un nouvel appareil peuvent être vérifiés par un gestionnaire de système par le biais de l'utilisation facultative de clés publiques ouvertement partagées par le nouvel appareil, et une clé privée asymétrique correspondante tenue secrète au sein du nouvel appareil.

Les PDU sont protégées en utilisant le chiffrement par défaut AES à blocs de 128 bits ou un autre chiffrement par blocs mandaté localement, en utilisant des modes cryptographiques normalisés. Des clés symétriques secrètes qui sont connues des entités en communication sont utilisées pour sécuriser la communication d'un appareil vers un autre.

4.6.13 Gestion de système

La présente norme inclut des fonctions pour gérer des ressources de communication sur chaque appareil individuel, ainsi que des ressources système qui ont un impact sur la performance de bout en bout. La gestion de système permet la prise en charge de la gestion politique de la configuration en temps d'exécution et elle effectue une surveillance et fournit des rapports sur la configuration, la performance, les conditions de défauts et le statut opérationnel. Les fonctions de gestion de système participent à des activités telles que:

- le rattachement à un réseau et départ d'un réseau pour un appareil;
- les rapports relatifs aux défauts qui se produisent dans le réseau;
- la configuration de communication;
- la configuration de la distribution d'horloges et le réglage du temps système;
- la surveillance d'appareil;
- la surveillance et l'optimisation de performances.

La gestion de sécurité du système travaille conjointement avec la fonction de gestion du système et, potentiellement, avec des systèmes externes de sécurité pour permettre l'exploitation sécurisée du système.

Toutes les fonctions de gestion sont accessibles à distance par l'intermédiaire de la passerelle.

4.6.14 Processus applicatif utilisant des objets normalisés

Le processus applicatif de la présente norme est représenté sous la forme d'un objet normalisé qui contient un ou plusieurs composants en communication puisant dans un ensemble d'objets d'application définis normalisés. Ces objets fournissent le stockage pour les données d'un processus applicatif et accèdent à celles-ci.

Le fait de définir des objets normalisés fournit d'une manière définitive une représentation des capacités d'une application distribuée, permettant de ce fait à des mises en œuvre indépendantes d'interfonctionner. Des objets sont définis pour permettre non seulement l'interaction parmi des appareils de terrain, mais aussi l'interfonctionnement avec des systèmes hôtes différents.

Les objets et services normalisés de la présente norme peuvent être utilisés pour établir un mapping direct des communications d'appareils de terrain héritées existantes à des objets normalisés et à des services de communication de sous-couche d'application, fournissant ainsi un moyen d'adapter des appareils hérités pour communiquer sur le réseau WISN.

4.6.15 Tunnellisation

Les protocoles natifs définis par la présente norme permettent à des appareils d'encapsuler des PDU étrangères et de transporter ces PDU étrangères, par le réseau WISN, vers un appareil de destination au sein du WISN, qui est habituellement une passerelle vers des protocoles hérités. Ce mécanisme d'encapsulation est appelé "tunnellisation". L'application réussie de la tunnellisation dépend de la bonne qualité avec laquelle les exigences techniques du protocole étranger (temporisation, latence, etc., par exemple) sont respectées par l'instanciation du WISN.

5 Système

5.1 Généralités

Dans la présente norme, un système est défini pour avoir un axe application et adresser des applications et leurs besoins. Les réseaux, d'autre part, ont un axe communication et sont consacrés à la tâche de communication d'un appareil à un autre. Pour les besoins de la présente norme, un réseau est un composant d'un système plus grand.

L'Article 5 décrit comment les diverses couches de protocoles et fonctions de la présente norme travaillent ensemble pour former un système qui atteint les buts de la présente norme. Spécifiquement, l'Article 5 décrit les aspects système des appareils, réseaux, suite de protocoles, flux de données, une base de temps partagée, et le besoin des applications pour les révisions de firmware.

5.2 Appareils

5.2.1 Généralités

Les appareils mettent en œuvre une combinaison des couches de protocoles, y compris habituellement une PhLE, une DLE, une NLE, une TLE et une ALE, et peut inclure des fonctions telles que le rôle de gestionnaire de système, le rôle de gestionnaire de sécurité, un ou plusieurs rôles de passerelle, et la prise en charge de configuration d'autres appareils dans le réseau.

NOTE Seuls les comportements de système sont spécifiés à l'Article 5.

5.2.2 Interfonctionnabilité d'appareils

L'interfonctionnabilité d'appareils est l'aptitude d'appareils issus de plusieurs fournisseurs à communiquer et maintenir le réseau complet. L'interfonctionnabilité d'appareils requiert la maîtrise sur les diverses options, valeurs de réglage de configuration et capacités de l'appareil:

a) **Options**: Pour permettre à tous les appareils d'interfonctionner, et d'interopérer dans un domaine restreint d'application, indépendamment des options mises en œuvre (celles définies dans la présente norme), les appareils doivent être capables de désactiver (c'est-à-dire de ne pas utiliser) toute option qui n'est pas obligatoire pour le ou les rôles configurés de l'appareil, tel que spécifié dans les profils de rôles à l'Annexe B.

b) **Valeurs de réglage de configuration**: Le gestionnaire de système est responsable de configurer les appareils du WISN et les rôles mis en œuvre par les appareils du WISN. Le gestionnaire de système est décrit à l'Article 6. Certains aspects de configuration sont décrits à l'Annexe D.

c) **Capacités**: Il existe des capacités minimales qui doivent être remplies pour les appareils en fonction de leur rôle dans le système. L'Annexe B définit les capacités de base exigées pour tous les appareils.

5.2.3 Profils

Un profil peut être décrit comme étant une tranche verticale à travers les couches de protocoles. Il définit les options dans chaque couche de protocoles qui sont obligatoires pour le profil en question. Il définit également les configurations et les plages de paramètres pour chaque protocole. Le concept de profil est utilisé pour réduire le risque de problèmes d'interfonctionnabilité et d'interopérabilité d'appareils entre les produits de différents fabricants. L'interopérabilité dans les zones figurant hors du domaine d'application de la présente norme exige soit l'utilisation de profils allant au-delà de la présente norme, soit d'autres dispositions complémentaires à la présente norme

Un profil de rôles est défini comme représentant les capacités de base, y compris toutes les caractéristiques facultatives, valeurs de réglages et configurations, qui sont exigées d'un appareil pour accomplir correctement le rôle en question. Les rôles sont définis en 5.2.6.2.

5.2.4 Qualité de service

Une application au sein d'un appareil est censée connaître le niveau du service qui est nécessaire pour son fonctionnement correct. Le niveau de la qualité du service (QoS) est convenu par un contrat entre le gestionnaire de système et l'appareil demandeur. Lorsque l'application au sein d'un appareil désire communiquer à un certain niveau de QoS, elle envoie une demande au gestionnaire de système lui notifiant qu'elle souhaite communiquer avec une destination spécifique et qu'elle désire un niveau donné de QoS. Ce niveau désiré de QoS est indiqué par une priorité désirée de contrat et de message. En outre, un certain niveau de fiabilité, une périodicité, une phase, et une date limite pour les messages périodiques, et une fréquence de salve de court terme, une fréquence de salve de long terme, et le nombre maximal de demandes en cours pour des client/serveur, peuvent être indiqués. Voir 6.3.11.2.7 pour des informations spécifiques relatives à la QoS.

5.2.5 Applicabilité mondiale d'appareil

5.2.5.1 Généralités

La présente norme est conçue pour prendre en charge le fonctionnement au sein d'une zone géographique fixe qui fonctionne sous des réglementations uniformes. A ce titre, elle est censée prendre en charge le fonctionnement dans la bande de 2,4 GHz n'importe où dans le monde (voir le débat en 9.1.15.6). Par exemple, l'Annexe V explique comment les systèmes et appareils peuvent être configurés pour satisfaire aux contraintes réglementaires de l'UE.

La présente norme est également conçue pour prendre en charge les systèmes d'automation sans fil fonctionnant sur des plateformes mobiles, telles que les navires marins (par exemple, les navires porte-conteneurs et les navires-citernes de produits pétrochimiques) et les trains, qui peuvent se déplacer entre des régions géographiques (des pays, par exemple) où s'appliquent des réglementations différentes voire conflictuelles. Par exemple, un navire porte-conteneurs serait habituellement soumis à des réglementations locales lorsqu'il est au port, et pourrait donc avoir des exigences de conformité différentes lorsqu'il est à Rotterdam de celles en vigueur dans la Baie de Tokyo, car les réglementations qui s'appliquent aux systèmes sans fil lorsqu'ils fonctionnent sous la juridiction de l'UE diffèrent de celles qui s'appliquent lorsqu'ils fonctionnent sous juridiction japonaise.

Les règlements des radiocommunications exigent souvent des appareils de fonctionner à des niveaux contraints de puissance, et ce, à tout moment, y compris pendant la configuration par liaison radio. Certaines limites de densité EIRP et EIRP identifiées sont: 10 mW/MHz (Japon); 10 dBm (Chine); 10 dBm, 10 mW/MHz et 20 dBm (UE); 36 dBm avec au maximum un gain d'antenne de 6 dBi (US FCC).

Dans certains pays, tels que la France, les niveaux d'émission sur certaines voies peuvent avoir besoin d'être affaiblis. Dans d'autres pays, tels que la Corée, le nombre et la gamme de voies ont besoin d'être contraints.

5.2.5.2 Fonctionnement au sein d'un régime de réglementation fixe

Le champ dlmo.CountryCode, décrit en 9.1.15.6, est utilisé pour spécifier le régime de réglementation. Il peut également être utilisé pour spécifier un certain nombre de contraintes de réglementation dérogatoires.

Ce champ inclut un mécanisme "autobloquant" qui permet de mettre ce champ à une valeur de sorte que le champ complet ne puisse pas être modifié pendant que l'appareil est opérationnel. Une fois la valeur définie, seule la reconfiguration de l'appareil, comme après une réparation ou un transfert de propriété, est capable de désactiver ce verrou.

Cette caractéristique "mettre et oublier" a été incluse pour prendre en charge des régimes de réglementation, tels que certains au sein de l'UE, où aucune méthode opérationnelle ne peut déroger aux limites d'émission RF établies par réglementation. Néanmoins, la caractéristique est fournie d'une manière qui prend également en charge la réparation et la revente d'appareils dans d'autres juridictions de réglementation, dont les exigences pourraient être différer de celles de la juridiction dans laquelle l'appareil a été déployé précédemment.

5.2.5.3 Fonctionnement sur une plateforme qui se déplace entre des régimes de réglementation

Certains systèmes d'automation sans fil peuvent être situés sur une plateforme mobile telle qu'un navire porte-conteneurs ou navire-citerne de produits pétrochimiques qui se déplace entre des régimes de réglementation, en fonctionnant temporairement dans chacun de ceux-ci. Cette transition entre régimes peut se produire rapidement, comme lorsqu'un train franchit une frontière, ou lentement, comme lorsqu'un bateau passe des eaux nationales aux eaux internationales.

La présente norme est conçue pour prendre en charge le fonctionnement de systèmes sans fil dans de telles situations de transport mobile, en fournissant un moyen par lequel un seul paramètre d'équipement peut être changé dans chaque appareil, par exemple par une action synchronisée téléchargée à l'avance vers chaque appareil, pour conduire tous les appareils affectés à changer de régimes de réglementation, après quoi leurs fonctionnements seront contraints par les réglementations des régimes de réglementation nouvellement pénétrés.

NOTE Un tel changement des caractéristiques d'émission sans fil sera souvent accompagné d'un changement des programmes de liaisons, par exemple, pour utiliser plus ou moins de routeurs dans un chemin à sauts multiples, ou pour avoir à disposition plus d'intervalles de temps pour la répétition des transactions qui ont été abandonnées en raison de l'activité détectée par le mécanisme LBT dans la voie, pour mieux faire concorder le fonctionnement du système avec les exigences plus strictes (ou assouplies) du nouveau régime de réglementation.

5.2.6 Description d'appareil

5.2.6.1 Généralités

Dans la présente norme, les appareils constituent le mode de réalisation physique des comportements, des valeurs de réglage de configuration et des capacités qui sont nécessaires pour mettre en œuvre et exploiter un réseau. Il existe un grand nombre de différents types d'appareils selon l'application, l'environnement, et leur fonction dans le réseau. Pour pouvoir décrire complètement le comportement nécessaire de réseau sans définir des mises en œuvre spécifiques d'appareil, la présente norme définit des rôles, des couches de protocoles, et un moyen de terrain dont les appareils peuvent être le mode de réalisation.

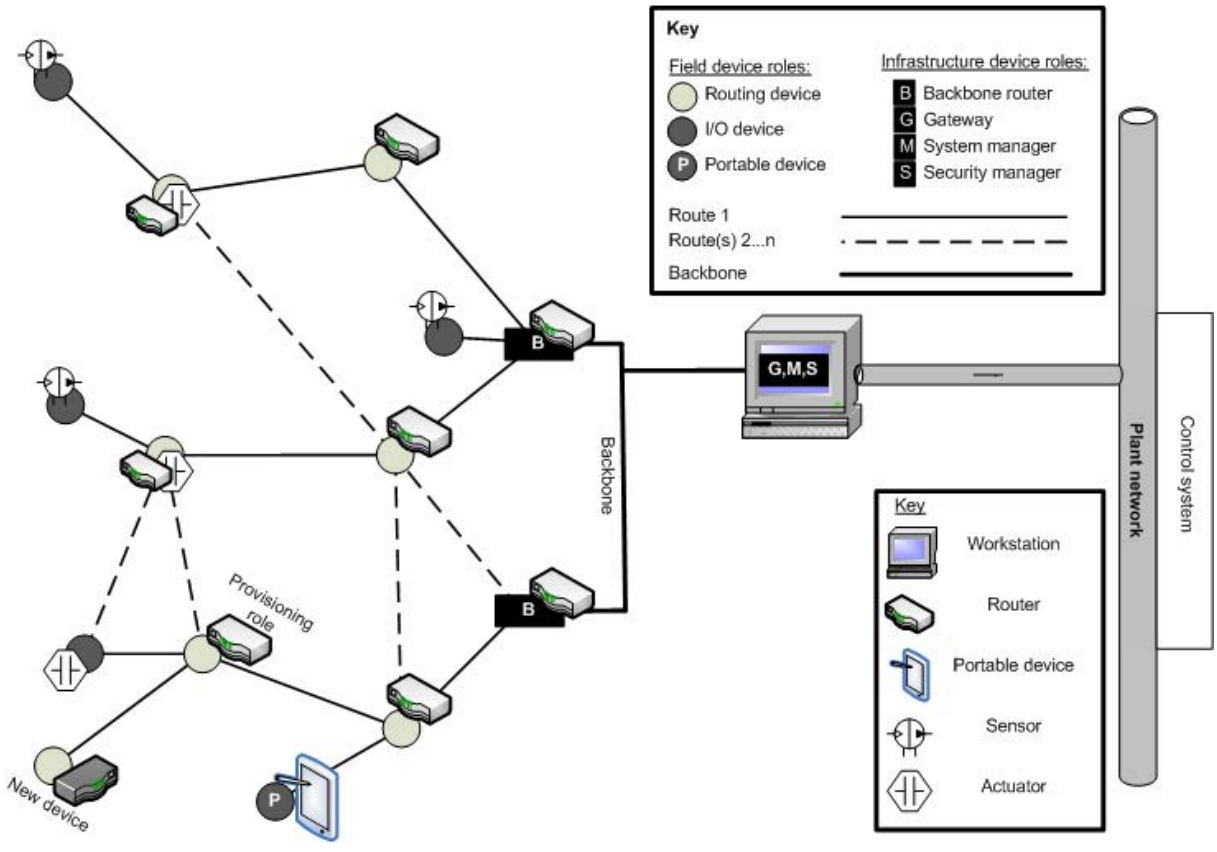
Un rôle définit un ensemble de fonctions et de capacités. La présente norme définit tous les rôles nécessaires au fonctionnement correct du réseau, y compris le gestionnaire de système, le gestionnaire de sécurité, la passerelle, le routeur dorsal, la source de temps système, la

configuration, le routeur, et l'appareil E/S. Tous les appareils conformes à la présente norme doivent mettre en œuvre au moins un rôle; cependant, un appareil peut mettre en œuvre plusieurs rôles. Un appareil mettant en œuvre un rôle doit mettre en œuvre toutes les fonctions exigées pour ce rôle en 5.3.

Les couches de protocoles décrivent les comportements exigés. Tous les appareils ne sont pas tenus mettre en œuvre toutes les couches de protocoles définies dans la présente norme. Par contre, tous les appareils conformes à la présente norme doivent mettre en œuvre les couches réseau et transport en plus de la fonctionnalité de DMAP telle que décrite en 6.2. Chaque appareil doit contenir une fonction de gestion d'appareil et une fonction de gestion de sécurité d'appareil qui coopèrent avec les processus du système pour permettre une gestion sécurisée des ressources d'un appareil et l'utilisation par l'appareil des ressources de système.

Un moyen de terrain est représenté au sein d'un appareil par une combinaison d'une PhLE et d'une DLE, toutes les deux telles que décrites dans la présente norme. Alors que tous les appareils n'ont pas nécessairement besoin de mettre en œuvre un moyen de terrain, tout appareil qui met en œuvre les rôles E/S, de routage ou de routage dorsal doit prendre directement en charge au moins un moyen de terrain tel que spécifié par la présente norme.

La Figure 4 montre la distinction entre appareils physiques (par exemple, tels que fournis par un fabricant) et les rôles que ces appareils peuvent assumer.



Légende

Anglais	Français
New device	Nouvel appareil
Provisioning role	Rôle de configuration
Backbone	Dorsale
Key	Clé
Field device roles	Rôles d'appareil de terrain
Routing device	Appareil de routage

Anglais	Français
I/O device	Appareil E/S
Portable device	Appareil portatif
Infrastructure device roles	Rôles d'appareil d'infrastructure
B Backbone router	B Routeur dorsal
G Gateway	G Passerelle
M System manager	M Gestionnaire système
S Security manager	S Gestionnaire de sécurité
Route 1	Chemin 1
Route(s) 2…n	Chemin(s) 2…n
Backbone	Dorsale
Key	Clé
Workstation	Station de travail
Router	Routeur
Portable device	Appareil portatif
Sensor	Capteur
Actuator	Actionneur
Control system	Système de commande
Plant network	Réseau d'installation

Figure 4 – Appareils physiques versus rôles

La Figure 4 montre un réseau représentatif et pourtant complet conforme à la présente norme. Ce réseau contient plusieurs types d'appareils, y compris des capteurs, des actionneurs, des routeurs, un calculateur à main et une station de travail. Comme montré à la Figure 4, chacun de ces appareils peut assumer différents rôles au sein du réseau. Dans cet exemple:

- la station de travail a assumé les rôles de passerelle, de gestionnaire de système et de gestionnaire de sécurité, ces rôles étant respectivement décrits en 5.2.6.10, en 5.2.6.11 et en 5.2.6.12;
- deux appareils ont assumé le rôle des routeurs dorsaux (décrit en 5.2.6.9) alors que sept autres appareils ont assumé le rôle de routeurs (décrit en 5.2.6.7);
- trois capteurs, un actionneur et un ordinateur portable ont assumé le rôle singulier d'un appareil E/S (5.2.6.6);
- le routeur en bas à gauche de la Figure 4 a assumé un rôle de configuration, tel que décrit en 5.2.6.8, et configurera le nouvel appareil en train d'être introduit; et
- deux appareils actionneurs ont assumé les rôles à la fois de routeur et de E/S.

NOTE 1 Bien que la Figure 4 montre l'utilisation d'un réseau dorsal, la fonctionnalité du réseau dorsal n'est pas spécifiée dans la présente norme.

NOTE 2 Les appareils physiques et les rôles montrés à la Figure 4 sont censés être seulement des exemples.

5.2.6.2 Moyen de terrain

5.2.6.3 Généralités

La présente norme définit un moyen de terrain spécifique, le Type A. Un type de moyen de terrain définit le protocole pour la PhL et la DL inférieure (à savoir la MAC). Les futures révisions de la présente norme peuvent prendre en charge plusieurs types de moyens de terrain.

5.2.6.4 Type A

Le moyen de terrain de Type A comprend la PhL et la DL telles que spécifiées par l'Article 8 et l'Article 9 de la présente norme.

Les appareils mettant en œuvre le moyen de terrain de Type A et la DL doivent mettre en œuvre valeurs de réglage pour le silence radio (Radio Silence). La configuration Radio Silence est utilisée pour empêcher la radio d'émettre pendant des périodes inadéquates, telles que quand l'émission est peu sûre ou quand des réglementations interdisent les émissions radio. Les valeurs de réglage de la configuration Radio Silence sont définies en 9.1.15.4.

5.2.6.5 Définitions des rôles

5.2.6.6 Entrée/sortie

Un appareil avec le rôle E/S (entrée/sortie) doit fournir des données (de source) à d'autres appareils ou utiliser (consommer) des données en provenance de ceux-ci (et peut aussi bien fournir qu'utiliser des données) et doit avoir au moins un objet processus d'application utilisateur (UAP). Un appareil avec seulement un rôle E/S est un appareil qui a les caractéristiques minimales exigées pour participer à un réseau conforme à la présente norme. Le rôle E/S ne fournit de mécanisme de transmission de messages ou de routage pour aucun autre appareil. Cela permet la construction d'appareils avec la plus faible complexité et le potentiel pour une faible consommation d'énergie, car ils n'ont pas besoin de dépenser de l'énergie pour acheminer les messages d'autres appareils et ils ne sont pas non plus tenus d'accepter et de configurer de nouveaux appareils souhaitant rejoindre le réseau.

NOTE Une source de données fournit des données. Un actionneur serait un exemple d'un consommateur de données (c'est-à-dire: un puits), alors qu'un capteur fournirait des données (c'est-à-dire: une source).

Les appareils qui mettent en œuvre le rôle E/S doivent mettre en œuvre le moyen de terrain de Type A.

5.2.6.7 Routeur

Un appareil avec le rôle de routeur doit avoir une capacité de routage, doit agir comme un proxy ("un mandataire") et doit avoir une capacité de propagation d'horloges. Ces appareils peuvent fournir une extension de portée pour un réseau et une redondance de chemin et peuvent fournir différents niveaux de QoS sur une base message par message. Le gestionnaire de système peut désactiver les capacités de routage du rôle de routeur pour optimiser des exigences relatives aux performances du système telles que la latence des messages ou la consommation de batterie.

Les appareils qui mettent en œuvre le rôle de routeur doivent mettre en œuvre le moyen de terrain de Type A.

5.2.6.8 Configuration

Un appareil avec le rôle de configuration (appareil de configuration) doit être capable de configurer un appareil mis aux valeurs par défaut d'usine et doit mettre en œuvre l'objet service de configuration d'appareil (DPSO; voir Article 13). L'appareil de configuration insère les données de configuration exigées dans un appareil pour permettre à un appareil de rejoindre un réseau spécifique. Les appareils mettant en œuvre la PhL doivent être capables d'être configurés en utilisant l'interface physique définie. Cette capacité peut être désactivée (voir Article 13).

Les appareils qui mettent en œuvre le rôle de configuration doivent mettre en œuvre le moyen de terrain de Type A.

5.2.6.9 Routeur dorsal

Un appareil avec le rôle de routeur dorsal doit avoir une capacité de routage, par l'intermédiaire de la dorsale, et doit agir comme un proxy en utilisant la dorsale. Les routeurs dorsaux permettent aux réseaux externes de porter le protocole natif en encapsulant les PDU pour le transport. Cela permet à un réseau décrit par la présente norme d'utiliser d'autres

réseaux, y compris des réseaux ayant une plus longue portée ou de plus hautes performances.

Alors que les supports et les suites de protocole des réseaux dorsaux ne sont pas définis dans la présente norme, il est jugé que de nombreuses instanciations du routeur dorsal le seront avec les réseaux IP (protocole internet). Plusieurs de ces réseaux dorsaux peuvent se conformer à IPv4 par opposition au plus nouvel IPv6. L'Article 10 décrit comment une NPDU WISN reçue par un BBR à l'interface DLE WISN du BBR est convertie en une NDPU pleinement conforme à l'IPv6. Si l'interface dorsale du BBR met en œuvre IPv6, alors la NPDU peut simplement être acheminée en utilisant la norme IPv6. Si l'interface dorsale du BBR met en œuvre IPv4, alors le BBR doit prendre en charge l'utilisation de l'IETF RFC 2529 pour acheminer la NPDU à la dorsale IPv4.

Les appareils mettant en œuvre le rôle de routeur dorsal doivent mettre en œuvre le moyen de terrain de Type A en plus de l'interface réseau dorsale du BBR.

5.2.6.10 Passerelle

Un appareil avec le rôle de passerelle met en œuvre une interface du côté haut. Un exemple d'un GIAP interne prenant en charge une telle interface de côté haut est donné en Annexe U. La passerelle communique sur le WISN par l'accès natif et/ou la tunnellisation. Un tel appareil doit avoir un UAP. Le rôle de passerelle fournit une interface entre le WISN et le réseau d'installation, ou directement à une application finale sur un réseau d'installation. Plus généralement, une passerelle marque la transition entre les communications conformes à la présente norme et les autres communications et agit comme convertisseur de protocole entre une AE décrite par la présente norme et les autres AE. Il peut y avoir plusieurs passerelles dans un système.

5.2.6.11 Gestionnaire de système

Un appareil mettant en œuvre le rôle de gestionnaire de système doit mettre en œuvre le SMAP (6.3.2) et doit positionner l'arbre de source de temps.

Le gestionnaire de système est une fonction spécialisée qui régit le réseau, les appareils, et les communications. Le gestionnaire de système accomplit la commande politique de la configuration en temps d'exécution du réseau, effectue une surveillance et fournit des rapports sur la configuration, la performance, le statut opérationnel et fournit des services relatifs au temps.

Lorsque deux appareils ont besoin de communiquer, ils le font en utilisant un contrat. Un contrat est un accord entre le gestionnaire de système et un appareil dans le réseau qui implique l'allocation de ressources du réseau par le gestionnaire de système pour prendre en charge un besoin particulier de communication de cet appareil. Ce contrat est conclu entre les applications tant dans les appareils que dans le gestionnaire de système. Le gestionnaire de système assignera au contrat un identificateur de contrat, et l'application dans l'appareil utilisera le contrat pour des communications. Une application peut seulement demander la création, la modification, ou la résiliation d'un contrat. Il est de la seule responsabilité du gestionnaire de système de créer, maintenir, modifier et résilier le contrat.

Pour plus d'informations sur la gestion de système, voir Article 6.

5.2.6.12 Gestionnaire de sécurité

La fonction de gestion de sécurité de système, ou le gestionnaire de sécurité, est une fonction spécialisée qui travaille conjointement avec le gestionnaire de système et, potentiellement, des systèmes externes de sécurité pour permettre l'exploitation sécurisée du système. Le gestionnaire de sécurité est logiquement séparable du gestionnaire de système et, dans certains cas d'utilisation, sera résident sur un appareil séparé et dans un lieu séparé. Chaque système conforme à la présente norme doit avoir un gestionnaire de sécurité. Pour plus d'informations sur le gestionnaire de sécurité, voir Article 7.

NOTE Le protocole de transmission utilisé entre le gestionnaire de système et le gestionnaire de sécurité n'est pas défini par la présente norme parce que de tels composants sont habituellement fournis par un fournisseur sous la forme d'un jeu/d'une paire assorti(e).

Pour plus d'informations sur la fonctionnalité de gestion de sécurité, voir 7.7.

5.2.6.13 Source de temps système

Un appareil mettant en œuvre le rôle de source de temps système doit mettre en œuvre la source de base de temps pour le système. Un sens de temps est un aspect important de la présente norme; il est utilisé pour gérer le fonctionnement de l'appareil. La source de temps système fournit un sens de temps pour le système entier. Cela est décrit en plus de détails en 6.3.10.1.

Les appareils mettant en œuvre le rôle de source de temps système doivent mettre en œuvre les rôles E/S, de routeur, de routeur dorsal, de gestionnaire de système, ou de passerelle.

5.2.7 Adressage d'appareil

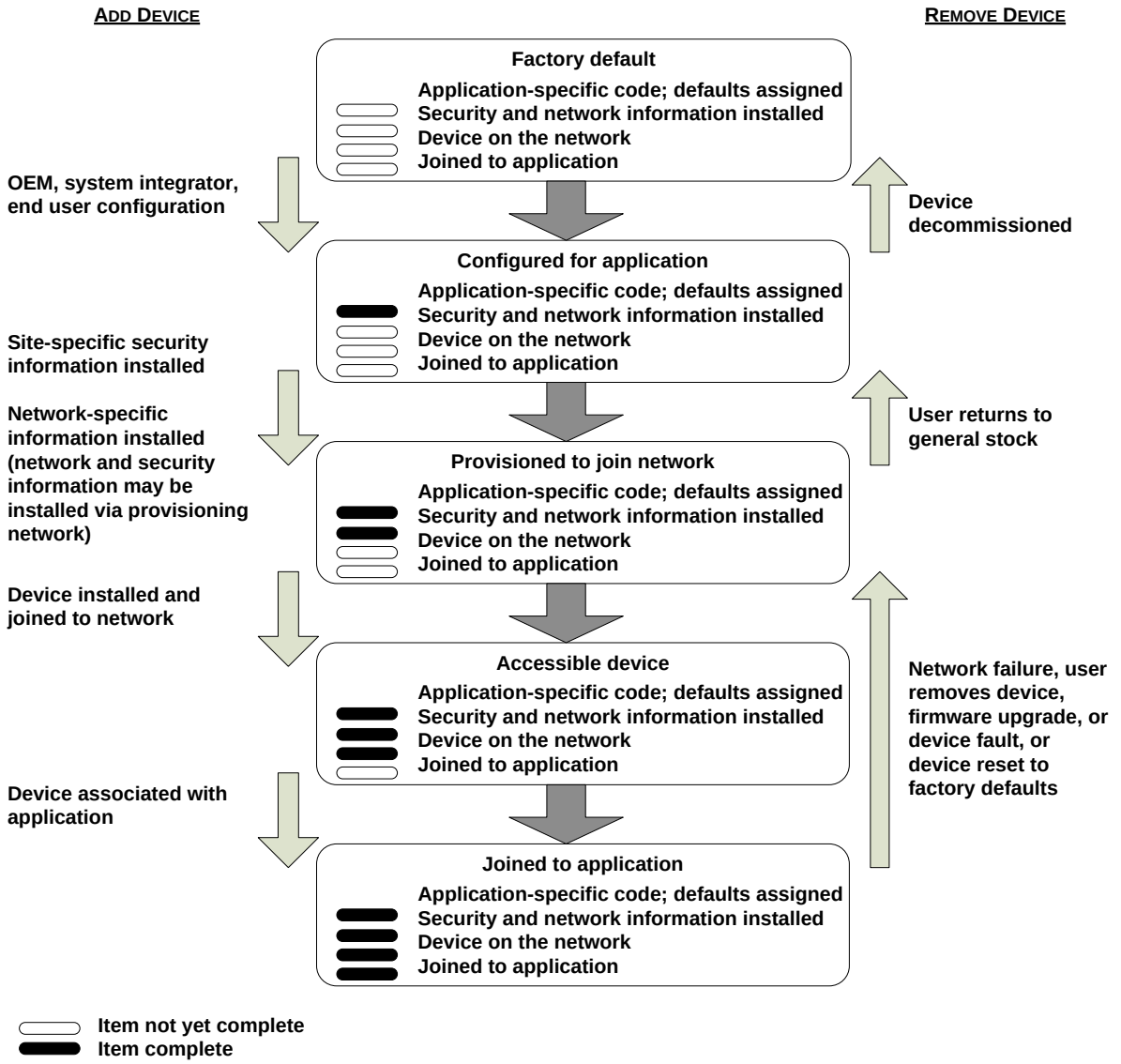
Chaque appareil qui met en œuvre le moyen de terrain de Type A doit avoir une adresse DL16Address de sous-réseau D qui lui est assignée en vue de l'adressage local. Chaque appareil doit avoir une adresse EUI64Address qui lui est propre. Voir Article 9 pour de plus amples informations.

Chaque appareil doit également avoir une adresseIPv6Address qui est assignée par le gestionnaire de système telle que décrite en 6.3.5. Le gestionnaire de système peut choisir d'assigner l'adresse IPv6Address comme une adresse logique pour maintenir la liaison d'ALE en cas de remplacement d'appareil. L'adresse IPv6Address peut être employée par l'application pour atteindre un appareil particulier au sein d'un système après que le processus de rattachement du sous-réseau se sera achevé. Voir Article 10 pour de plus amples informations.

5.2.8 Phases d'appareil

5.2.8.1 Généralités

Un appareil peut passer par plusieurs phases pendant sa durée de vie opérationnelle. Chacune de ces phases contient plusieurs états. Une représentation hypothétique des phases de la vie d'un appareil est montrée à la Figure 5. Voir la Figure 136 et la Figure 137 pour le détail normatif.



Légende

Anglais	Français
Add device	Ajouter appareil
Remove device	Retirer appareil
Factory default	Valeur d'usine par défaut
Application-specific code; defaults assigned	Code spécifique à l'application; valeurs par défaut assignées
Security and network information installed	Informations relatives à la sécurité et au réseau installées
Device on the network	Appareil sur le réseau
Joined to application	Rattaché à l'application
OEM, system integrator, end user configuration	OEM, intégrateur système, configuration d'utilisateur final
Device decommissioned	Appareil retiré du service
Configured for application	Configuré pour l'application
Application-specific code; defaults assigned	Code spécifique à l'application; valeurs par défaut assignées
Security and network information installed	Informations relatives à la sécurité et au réseau installées

Anglais	Français
Device on the network	Appareil sur le réseau
Joined to application	Rattaché à l'application
Site-specific security information installed	Informations relatives à la sécurité spécifiques au site installées
Network-specific information installed (network and security information may be installed via provisioning network)	Informations relatives à la sécurité spécifiques au réseau installées (les informations relatives au réseau et à la sécurité peuvent être installées par l'intermédiaire du réseau de configuration)
User returns to general stock	L'utilisateur retourne au stock général
Provisioned to join network	Configuré pour rejoindre le réseau
Application-specific code; defaults assigned	Code spécifique à l'application; valeurs par défaut assignées
Device installed and joined to network	Appareil installé et rattaché au réseau
Accessible device	Appareil accessible
Application-specific code; defaults assigned	Code spécifique à une application; valeurs par défaut assignées
Network failure, user removes device, firmware upgrade, or device fault, or device reset to factory defaults	Défaillance du réseau, l'utilisateur retire l'appareil, mise à niveau de firmware, ou panne d'appareil, ou réinitialisation de l'appareil aux valeurs d'usine par défaut
Device associated with application	Appareil associé à l'application
Joined to application	Rattaché à l'application
Application-specific code; defaults assigned	Code spécifique à l'application; valeurs par défaut assignées
Security and network information installed	Informations relatives à la sécurité et au réseau installées
Device on the network	Appareil sur le réseau
Joined to application	Rattaché à l'application
Item not yet complete	Elément pas encore complet
Item complete	Elément complet

Figure 5 – Représentation hypothétique des phases d'un appareil

Un appareil peut passer plusieurs fois par ces phases pendant qu'il est mis en service et utilisé, puis retiré du service et remis en service pour une application différente. Après avoir rejoint le réseau, les appareils doivent être capables de rapporter leur statut afin que les applications sachent si un appareil est accessible et s'il est rattaché à une application.

5.2.8.2 Valeur par défaut d'usine

Un appareil est considéré comme étant non configuré s'il n'a été configuré ou mis en service avec aucune information spécifique à une application ou spécifique à un réseau. Un appareil non configuré peut provenir d'un fabricant ou peut entrer dans un état non configuré à la suite de son retrait du service.

5.2.8.3 Configuré pour application

Un appareil est considéré comme étant configuré pour une application lorsqu'il a reçu sa propre programmation spécifique à une application et lorsque toutes les valeurs par défaut appropriées ont été ont été appliquées. Un appareil configuré pour application peut provenir d'un fabricant ou peut être fourni par un intégrateur de systèmes ou un autre revendeur à valeur ajoutée, en étant déjà configuré pour l'application prévue. Des mises à jour de programmes d'application en liaison radio peuvent avoir lieu, mais sont traitées par l'ALE de l'appareil.

5.2.8.4 Configuré pour rejoindre le réseau

Un appareil est configuré pour rejoindre le réseau lorsqu'il a obtenu les authentifiants de sécurité et les informations spécifiques au réseau qui sont appropriés. Un appareil entre habituellement dans cette phase lorsqu'il a été préparé pour l'installation dans une application d'automation. Souvent, l'appareil ne communiquera pas directement avec le gestionnaire de sécurité; à la place, le gestionnaire de système sert de relais pour toute la communication avec le gestionnaire de sécurité.

5.2.8.5 Appareil accessible

Un appareil est considéré comme étant accessible lorsqu'il a rejoint le réseau et a été authentifié par le gestionnaire de système. Un appareil accessible peut communiquer avec le gestionnaire de système.

5.2.8.6 Rattaché à une application

Dans cette phase, un objet d'application sur l'appareil peut envoyer ou recevoir de l'information à destination ou en provenance des objets d'application désirés sur des appareils homologues. Voir 7.4 pour plus d'infirmations sur le processus de rattachement du sous-réseau.

NOTE Les objets d'application dans deux appareils quelconques sur le réseau peuvent communiquer l'un avec l'autre. Se référer à l'Article 12 pour plus de détails.

5.2.9 Sources d'énergie d'appareil

La présente norme ne limite pas les types de sources d'énergie qu'un appareil peut utiliser. La norme permet le comportement de haut rendement énergétique d'un appareil qui adapte le fonctionnement de l'appareil pendant de longues périodes (cinq à dix ans, par exemple) en utilisant les accumulateurs appropriés.

Les types de sources d'énergie peuvent être groupés en cinq catégories:

- secteur;
- accumulateur limité (élément bouton, par exemple);
- accumulateur modéré (au plomb-bioxyde de plomb, par exemple);
- batterie rechargeable;
- environnemental ou récupérateur d'énergie.

Les appareils mettant en œuvre les rôles E/S ou de routeur peuvent être censés utiliser n'importe quelle catégorie de source d'énergie. Les rôles du gestionnaire de sécurité, de gestionnaire de système, de passerelle, et de routeur dorsal sont habituellement intensifs en performances; il convient donc que les appareils mettant en œuvre ces rôles aient des sources d'énergie plus capables, comme les catégories secteur ou accumulateur modéré.

Le statut de la source d'énergie des appareils est critique pour la gestion correcte du système. Tous les appareils doivent fournir au gestionnaire de système les informations relatives à l'alimentation en énergie. Ces informations peuvent être utilisées pour prendre des décisions de routage. Voir Article 9 pour plus d'informations.

5.3 Réseaux

5.3.1 Généralités

L'axe sur lequel les réseaux sont centrés est la communication d'un appareil vers un autre. Il existe de nombreux aspects de la capacité des réseaux à communiquer. Ces aspects incluent le réseau atomique (c'est-à-dire minimal ou irréductible), les topologies de réseau, les

relations des appareils au sein d'un réseau, la structure de la suite de protocoles, et le concept de temps partagé.

5.3.2 Réseau minimal

Un réseau minimal est un réseau avec la quantité minimale d'appareils mettant en œuvre le nombre minimal de rôles. Bien qu'un système minimal puisse être construit avec juste un gestionnaire de système et un gestionnaire de sécurité, un système minimum plus pratique inclut les rôles de gestionnaire de système, de gestionnaire de sécurité, de configuration, de source de temps système et d'E/S. Le gestionnaire de système et le gestionnaire de sécurité constituent deux rôles distincts; ils peuvent résider dans le même appareil ou être divisés entre deux appareils physiques. Un même appareil physique peut assumer plusieurs rôles. Par conséquent, un réseau minimal doit être constitué de deux appareils communiquant l'un avec l'autre, où un appareil met en œuvre les rôles de gestionnaire de système et de gestionnaire de sécurité; les rôles de configuration, de source de temps système et E/S sont mis en œuvre par l'un ou l'autre des appareils.

Un petit réseau représentatif de quatre appareils de terrain et d'un appareil d'infrastructure est montré à la Figure 6. Bien qu'un tel réseau soit atypique, il représente un petit système conforme. Dans ce réseau, un même appareil physique a assumé les rôles de passerelle, de gestionnaire de système et de gestionnaire de sécurité.

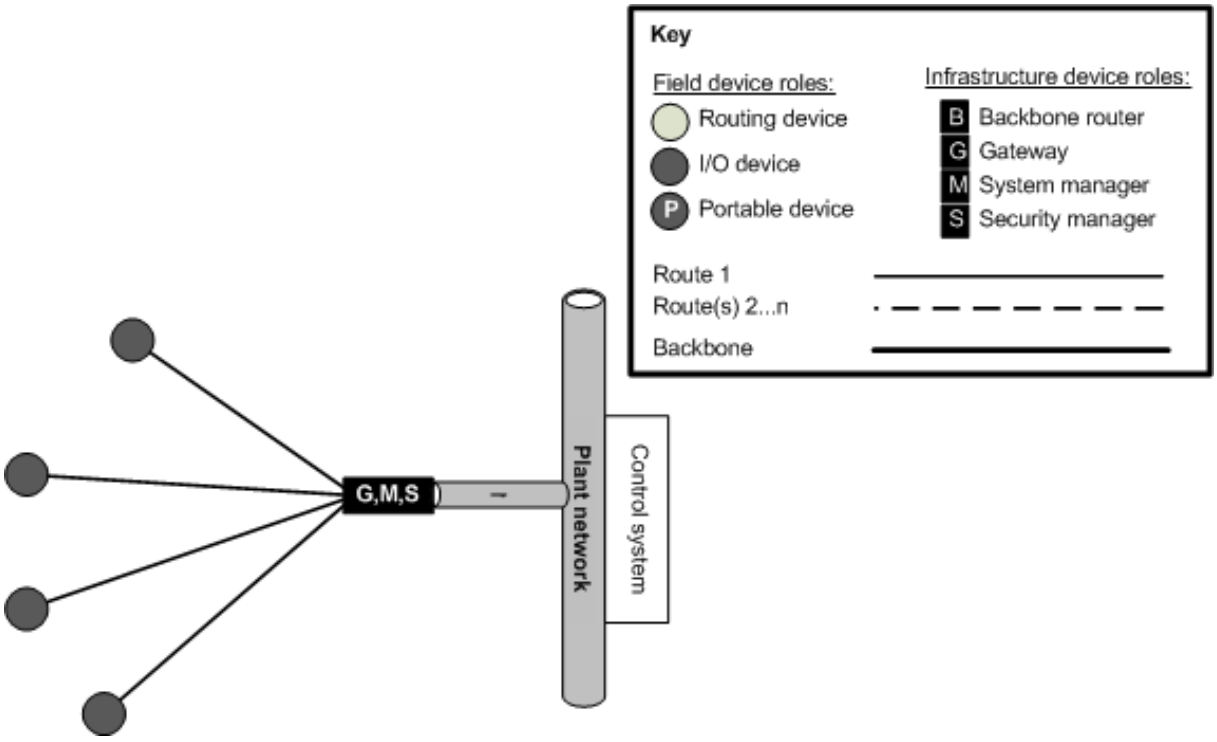
5.3.3 Topologies réseau de base prises en charge

5.3.3.1 Généralités

Les Figures 6 à 9 en 5.3.3 fournissent plusieurs exemples informatifs et illustrent la flexibilité de l'architecture d'un système. L'ensemble d'exemples n'est pas censé être exhaustif. Ces exemples sont présentés ici pour permettre seulement une meilleure compréhension éléments d'un système.

5.3.3.2 Topologie en étoile

La présente norme prend en charge une topologie en étoile simple, telle que montrée à la Figure 6.



Légende

Anglais	Français
Key	Légende
Field device roles	Rôles d'appareil de terrain
Routing device	Appareil de routage
I/O device	Appareil E/S
Portable device	Appareil portatif
Infrastructure device roles	Rôles d'appareil d'infrastructure
B Backbone router	B Routeur dorsal
G Gateway	G Passerelle
M System manager	M Gestionnaire de système
S Security manager	S Gestionnaire de sécurité
Route 1	Chemin 1
Route(s) 2...n	Chemin(s) 2...n
Backbone	Dorsale
Control system	Système de commande
Plant network	Réseau d'installation

Figure 6 – Topologie en étoile simple

Cette configuration de système peut donner la plus faible latence possible à travers la couche physique. Elle est très simple du point de vue de l'architecture, mais se limite à la portée d'un seul saut.

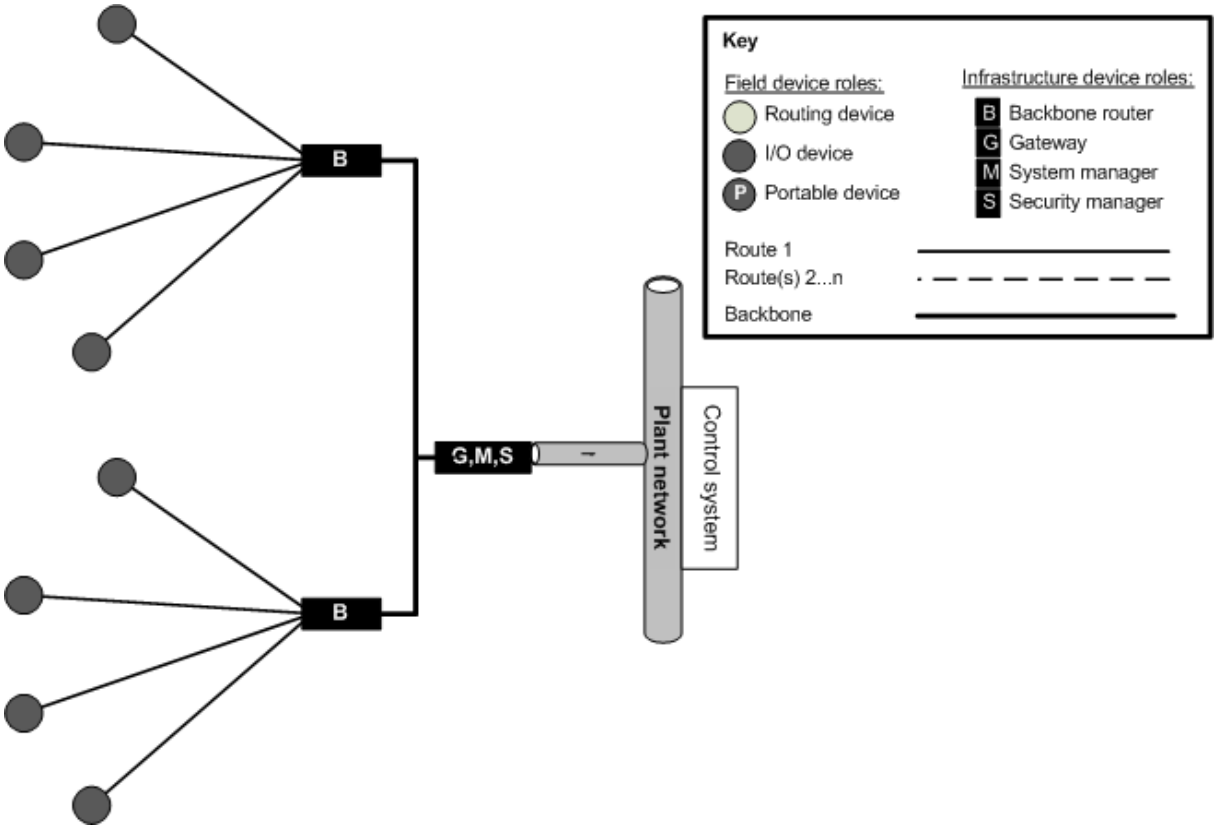
Dans les Figures 6 à 9 en 5.3.3, chaque zone étiquetée G,M,S représente un ensemble de trois rôles séparés combinés dans un seul appareil physique dans les réseaux relativement simples illustrés:

- une passerelle;
- un gestionnaire de système; et

- un gestionnaire de sécurité.

5.3.3.3 Topologie concentrateur-rayons

Le fait d'étendre le réseau en utilisant des routeurs dorsaux permet à l'utilisateur de construire un réseau concentrateur-rayons (ou réseau en étoile) (voir Figure 7) dans lequel les appareils sont placés en grappes autour de chaque routeur dorsal, fournissant l'accès à la dorsale à grande vitesse.



Légende

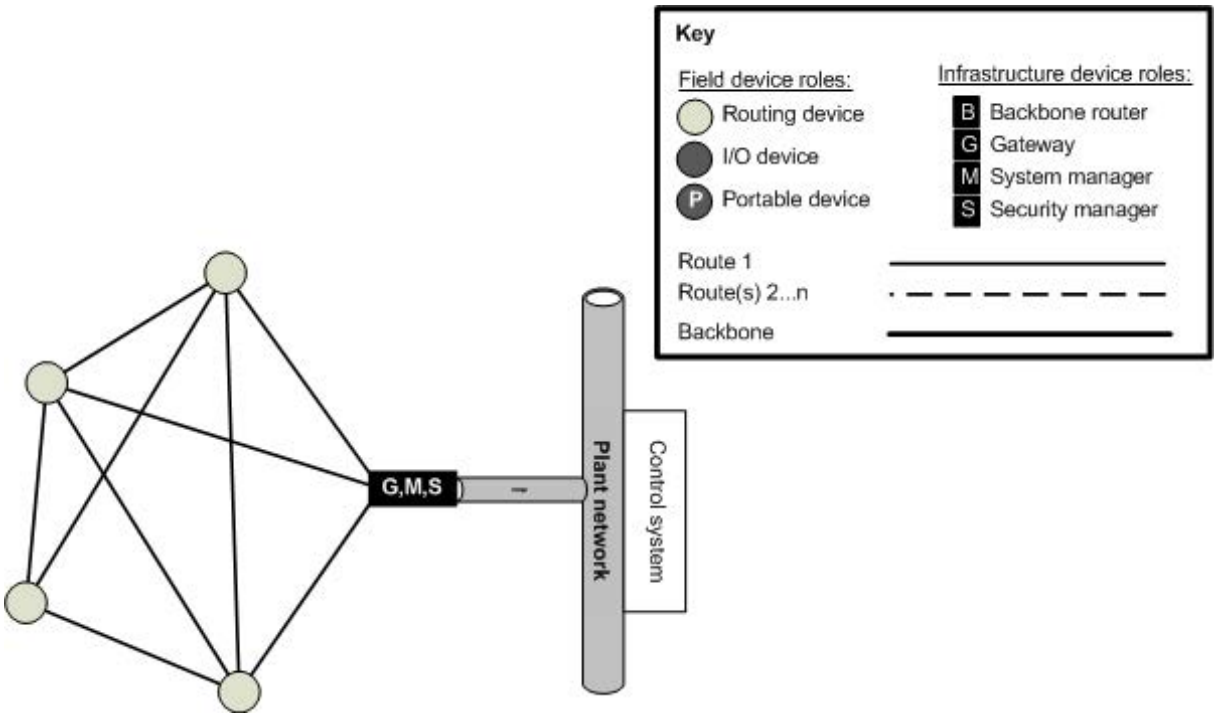
Anglais	Français
Key	Légende
Field device roles	Rôles d'appareil de terrain
Routing device	Appareil de routage
I/O device	Appareil E/S
Portable device	Appareil portatif
Infrastructure device roles	Rôles d'appareil d'infrastructure
B Backbone router	B Routeur dorsal
G Gateway	G Passerelle
M System manager	M Gestionnaire de système
S Security manager	S Gestionnaire de sécurité
Route 1	Chemin 1
Route(s) 2...n	Chemin(s) 2...n
Backbone	Dorsale
Control system	Système de commande
Plant network	Réseau d'installation

Figure 7 – Topologie concentrateur-rayons simple

Dans ce cas, la latence est légèrement dégradée par rapport à la topologie en étoile simple, mais le débit global peut augmenter, et dans de plus gros systèmes, la latence moyenne peut diminuer en raison des multiples tubes de données disponibles (un à travers chaque routeur dorsal). Bien que le réseau puisse s'étendre encore plus loin de la passerelle, il est néanmoins limité à la portée d'un seul saut autour d'un routeur dorsal.

5.3.3.4 Topologie maillée

La présente norme prend en charge les topologies de mise en réseau maillé, telles que montrées à la Figure 8.



Légende

Anglais	Français
Key	Légende
Field device roles	Rôles d'appareil de terrain
Routing device	Appareil de routage
I/O device	Appareil E/S
Portable device	Appareil portatif
Infrastructure device roles	Rôles d'appareil d'infrastructure
B Backbone router	B Routeur dorsal
G Gateway	G Passerelle
M System manager	M Gestionnaire de système
S Security manager	S Gestionnaire de sécurité
Route 1	Chemin 1
Route(s) 2...n	Chemin(s) 2...n
Backbone	Dorsale
Control system	Système de commande
Plant network	Réseau d'installation

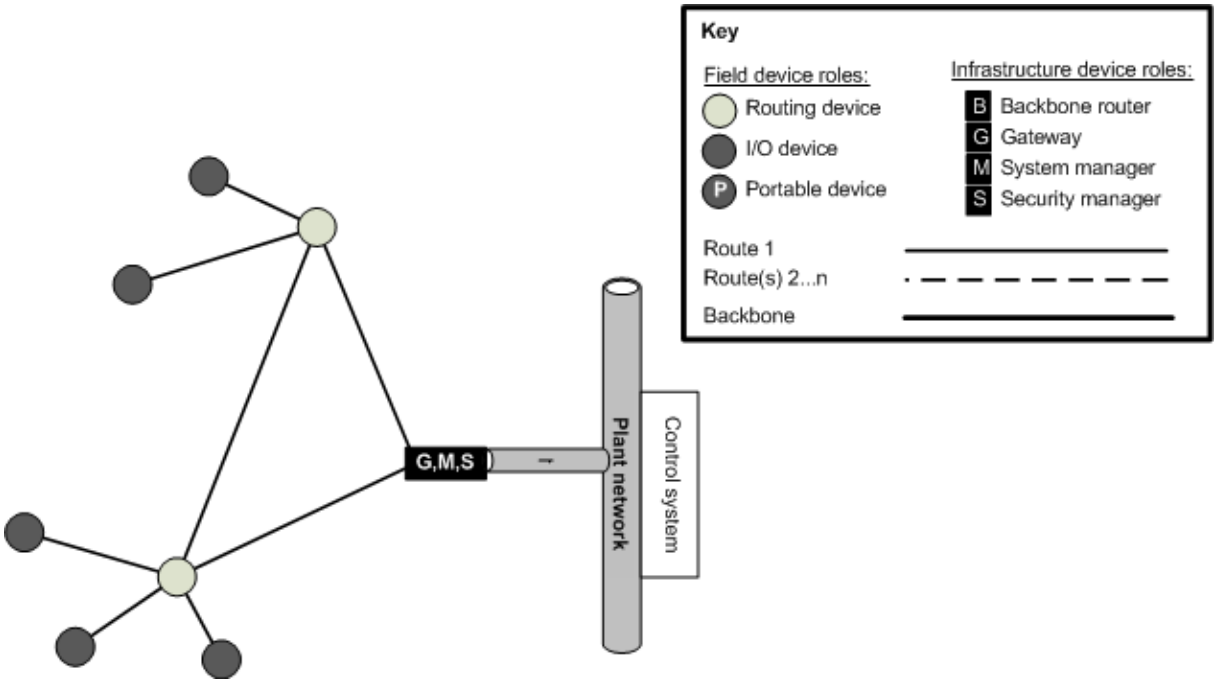
Figure 8 – Topologie maillée

Dans certains cas, le nombre d'itinéraires qu'un appareil peut prendre en charge peut être limité. La portée s'allonge à mesure que plusieurs sauts sont pris en charge. La latence est plus grande, mais peut être réduite au maximum par une programmation correcte des émissions. Le débit se dégrade à mesure que les ressources de l'appareil sont utilisées dans des messages répétitifs. La fiabilité peut être améliorée par l'utilisation d'une diversité de chemins.

Pour plus d'informations sur la topologie maillée, voir 9.1.14.

5.3.3.5 Topologie maillée en étoile

La combinaison de la topologie en étoile avec la topologie maillée est montrée à la Figure 9.



Légende

Anglais	Français
Key	Légende
Field device roles	Rôles d'appareil de terrain
Routing device	Appareil de routage
I/O device	Appareil E/S
Portable device	Appareil portatif
Infrastructure device roles	Rôles d'appareil d'infrastructure
B Backbone router	B Routeur dorsal
G Gateway	G Passerelle
M System manager	M Gestionnaire de système
S Security manager	S Gestionnaire de sécurité
Route 1	Chemin 1
Route(s) 2...n	Chemin(s) 2...n
Backbone	Dorsale
Control system	Système de commande
Plant network	Réseau d'installation

Figure 9 – Topologie en étoile-maillée simple

Cette configuration a l'avantage de limiter le nombre de sauts dans un réseau. Elle n'a pas la fiabilité ajoutée que la pleine mise en réseau maillé peut fournir.

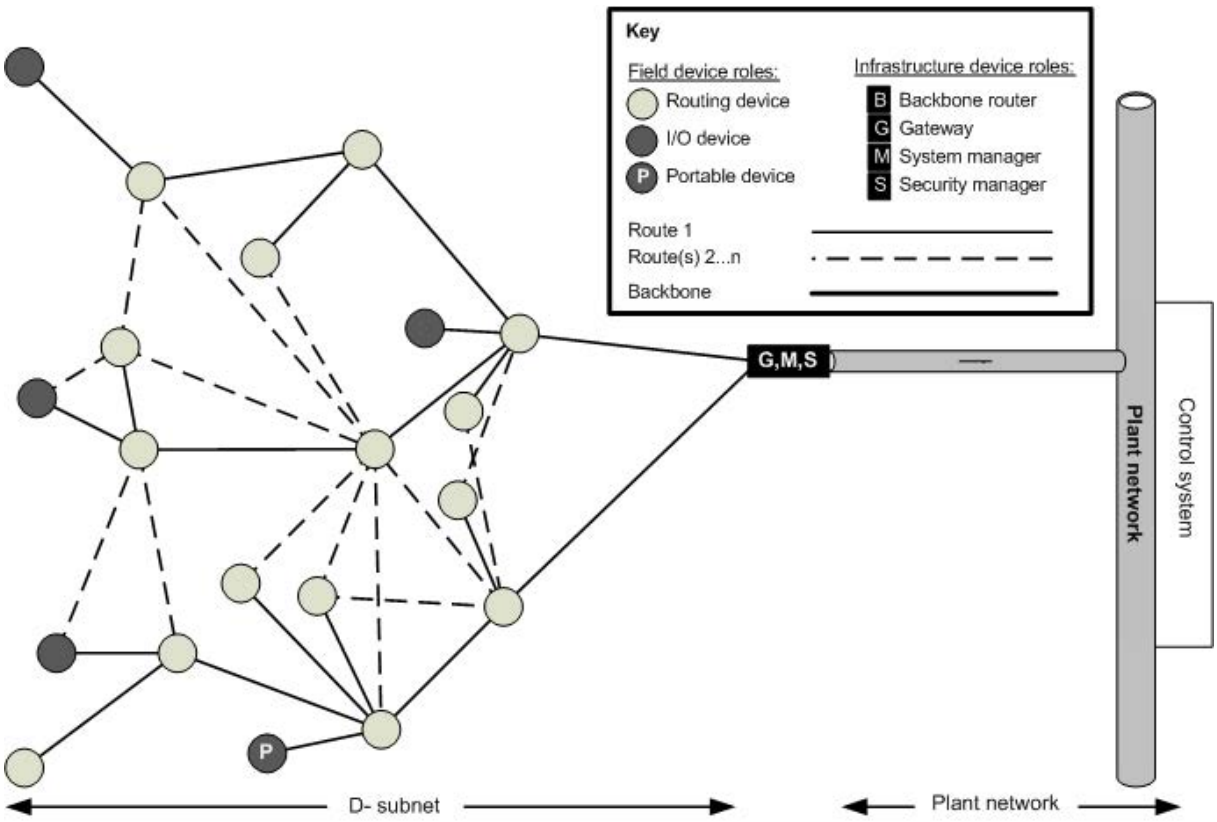
5.3.3.6 Combinaisons de topologies

La présente norme permet la combinaison de n'importe lesquelles des topologies mentionnées précédemment afin que puisse être construite une configuration qui satisfait le mieux aux besoins de l'application. Par exemple, des systèmes de surveillance qui couvrent de grandes zones physiques au sein d'une installation peuvent utiliser la topologie en étoile maillée ou une combinaison de topologies concentrateur-rayons et en étoile maillée, alors que certaines applications de contrôle où la latence est critique peuvent tirer bénéfice d'une topologie en étoile ou concentrateur-rayons. La flexibilité du système permet à toutes ces topologies de fonctionner en harmonie, sous n'importe quelle combinaison.

5.3.4 Configurations de réseau

5.3.4.1 Généralités

Le sous-réseau D dans la présente norme comprend un ou plusieurs groupes d'appareils sans fil, avec un gestionnaire de système partagé et (si applicable) une dorsale partagée. Alors qu'un sous-réseau D s'arrête au niveau du routeur dorsal (voir 5.5.6), le routage de réseau peut s'étendre dans la dorsale et le réseau d'installation. Un réseau complet inclut tous les sous-réseaux D connexes, ainsi que d'autres appareils reliés par l'intermédiaire de la dorsale, tels qu'une passerelle, un gestionnaire de système, ou un gestionnaire de sécurité. La Figure 10 et la Figure 11 montrent la distinction entre un sous-réseau D et un réseau.



Légende

Anglais	Français
Key	Légende
Field device roles	Rôles d'appareil de terrain
Routing device	Appareil de routage
I/O device	Appareil E/S
Portable device	Appareil portatif

Anglais	Français
Infrastructure device roles	Rôles d'appareil d'infrastructure
B Backbone router	B Routeur dorsal
G Gateway	G Passerelle
M System manager	M Gestionnaire de système
S Security manager	S Gestionnaire de sécurité
Route 1	Chemin 1
Route(s) 2…n	Chemin(s) 2…n
Backbone	Dorsale
Control system	Système de commande
Plant network	Réseau d'installation
D-subnet	Sous-réseau D

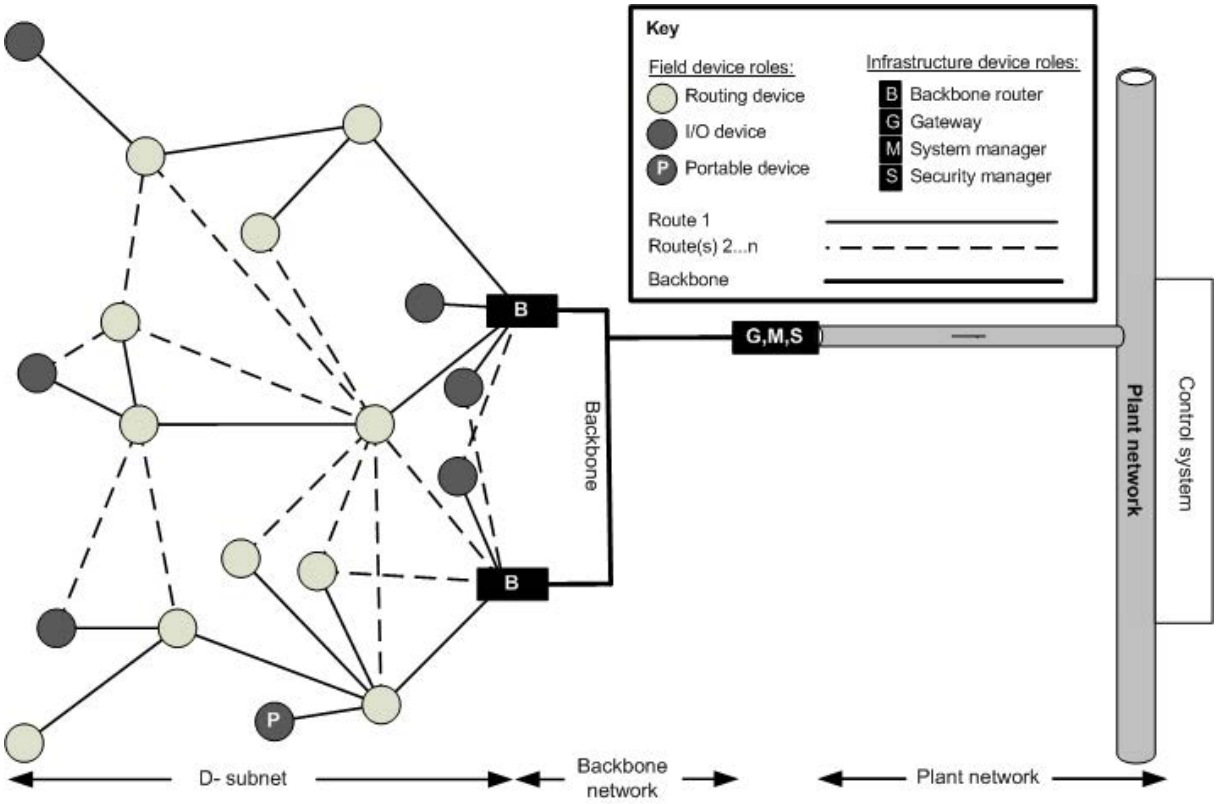
Figure 10 – Exemple où le réseau et le sous-réseau D se chevauchent

La Figure 10 montre un réseau simple constitué d'un ensemble d'appareils sans fil appelés un sous-réseau D et d'appareils complémentaires qui gèrent le sous-réseau D et le relient à d'autres réseaux. Dans la Figure 10, le réseau et le sous-réseau de D sont les mêmes.

Le sous-réseau D est composé d'appareils de routage et d'appareils E/S. Les lignes en traits pleins entre les appareils désignent le premier itinéraire établi entre les appareils, alors que les lignes pointillées désignent le deuxième itinéraire, le troisième itinéraire, et ainsi de suite. Des messages peuvent être acheminés en utilisant n'importe lequel des itinéraires connus.

Dans la Figure 11, le sous-réseau D inclut un ensemble d'appareils de terrain jusqu'aux routeurs dorsaux (boîtes étiquetées B). Les routeurs dorsaux utilisent des connexions à une dorsale de réseau pour réduire le nombre de sauts que les messages exigeraient autrement, ce qui peut améliorer la fiabilité, réduire la latence et étendre la couverture du réseau.

Le réseau dans la Figure 11 inclut le sous-réseau D, ainsi que la dorsale et une passerelle, un gestionnaire de système, et un gestionnaire de sécurité, qui sont colocalisés sur la dorsale.



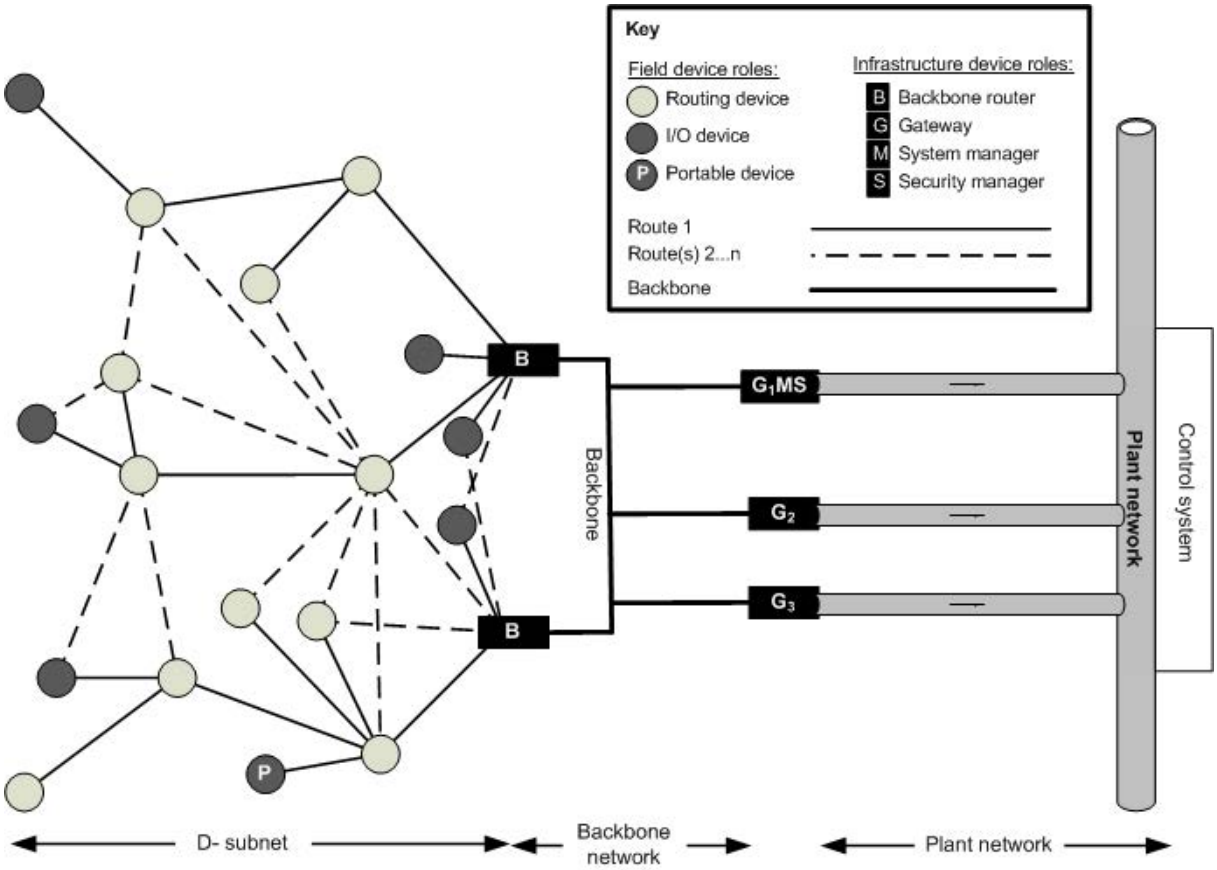
Légende

Anglais	Français
Key	Légende
Field device roles	Rôles d'appareil de terrain
Routing device	Appareil de routage
I/O device	Appareil E/S
Portable device	Appareil portatif
Infrastructure device roles	Rôles d'appareil d'infrastructure
B Backbone router	B Routeur dorsal
G Gateway	G Passerelle
M System manager	M Gestionnaire de système
S Security manager	S Gestionnaire de sécurité
Route 1	Chemin 1
Route(s) 2…n	Chemin(s) 2…n
Backbone	Dorsale
Control system	Système de commande
Plant network	Réseau d'installation
Backbone	Dorsale
D-subnet	Sous-réseau D
Backbone network	Réseau dorsal

Figure 11 – Exemple où le réseau et le sous-réseau D diffèrent

5.3.4.2 Passerelles multiples – Redondance et fonctions complémentaires

La Figure 12 montre une configuration physique différente avec trois appareils passerelles. L'un des appareils passerelles met également en œuvre les fonctions de gestionnaire de système et de gestionnaire de sécurité.



Légende

Anglais	Français
Key	Légende
Field device roles	Rôles d'appareil de terrain
Routing device	Appareil de routage
I/O device	Appareil E/S
Portable device	Appareil portatif
Infrastructure device roles	Rôles d'appareil d'infrastructure
B Backbone router	B Routeur dorsal
G Gateway	G Passerelle
M System manager	M Gestionnaire de système
S Security manager	S Gestionnaire de sécurité
Route 1	Chemin 1
Route(s) 2…n	Chemin(s) 2…n
Backbone	Dorsale
Control system	Système de commande
Plant network	Réseau d'installation
Backbone	Dorsale
D-subnet	Sous-réseau D
Backbone network	Réseau dorsal

Figure 12 – Réseau avec plusieurs passerelles

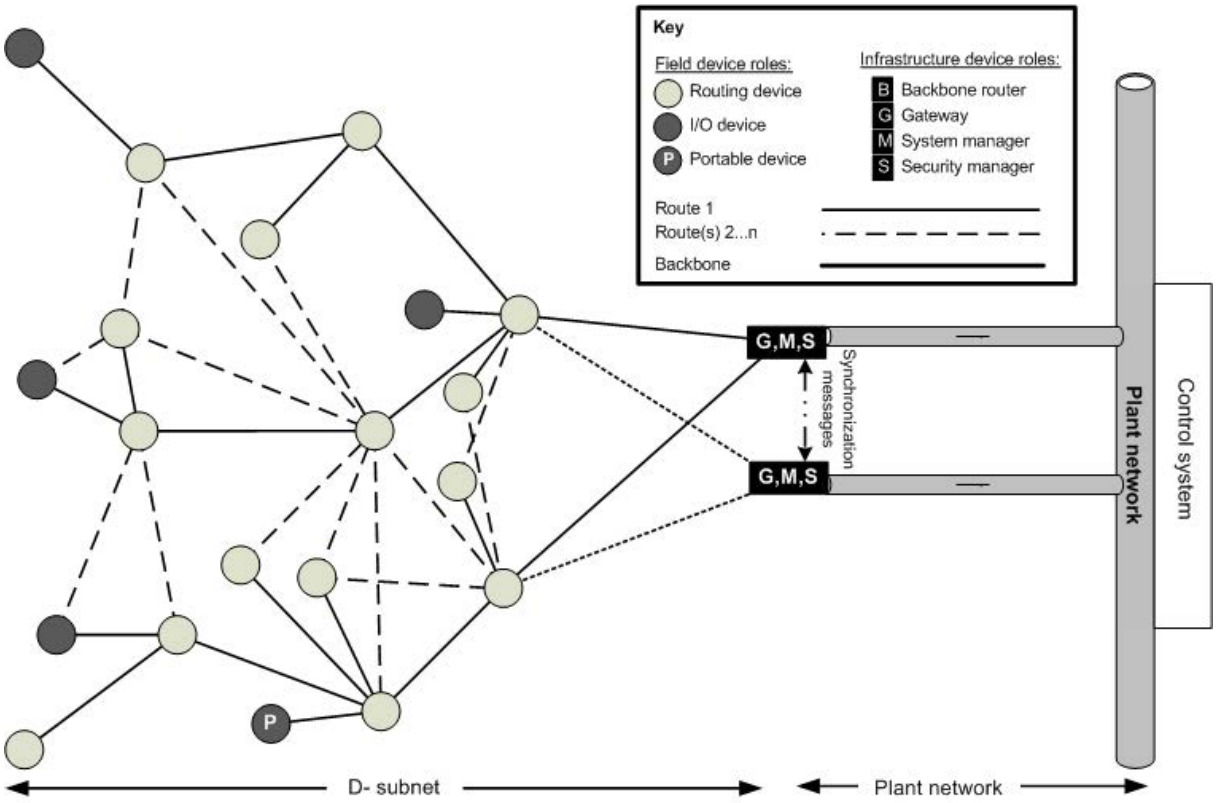
Les appareils passerelles peuvent être identiques (c'est-à-dire: reproduits en miroir, pour la redondance) ou uniques, par exemple, avec chaque passerelle mettant en œuvre une

application de logiciel pour traiter les communications entre une classe particulière d'appareil et un système de contrôle rattaché au réseau d'installation.

5.3.4.3 Plusieurs passerelles – Désignation d'une passerelle comme secours

NOTE La présente norme ne définit ni la fonctionnalité d'un passerelle de secours ni les mécanismes pour la synchronisation des passerelles de secours.

La Figure 13 est semblable à la Figure 10, mais avec un second appareil G,M,S (passerelle, gestionnaire de système, et gestionnaire de sécurité).



Légende

Anglais	Français
Key	Clé
Field device roles	Rôles d'appareil de terrain
Routing device	Appareil de routage
I/O device	Appareil E/S
Portable device	Appareil portatif
Infrastructure device roles	Rôles d'appareil d'infrastructure
B Backbone router	B Routeur dorsal
G Gateway	G Passerelle
M System manager	M Gestionnaire de système
S Security manager	S Gestionnaire de sécurité
Route 1	Chemin 1
Route(s) 2…n	Chemin(s) 2…n
Backbone	Dorsale
Control system	Système de commande
Plant network	Réseau d'installation
Synchronization messages	Messages de synchronisation

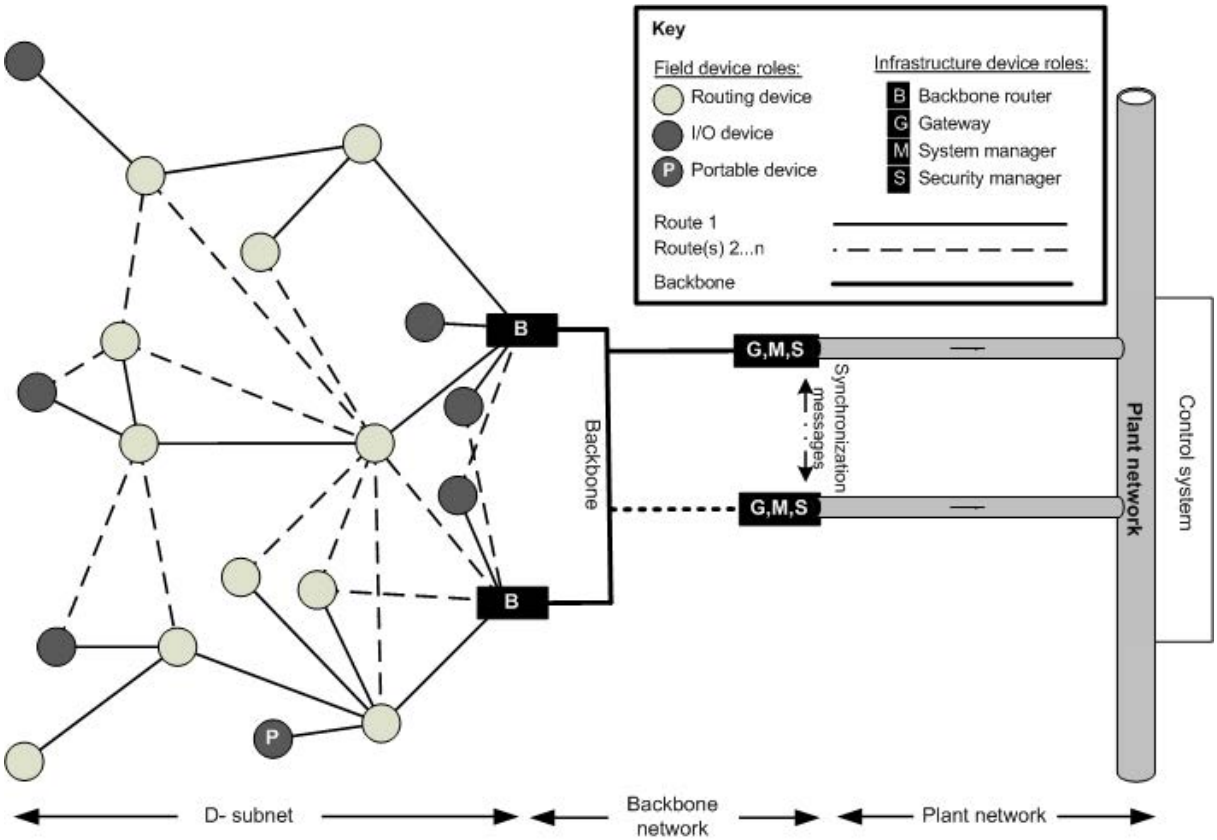
Anglais	Français
D-subnet	Sous-réseau D

Figure 13 – Réseau de base avec passerelle de secours

Les deux appareils G,M,S offrent une fonctionnalité identique et peuvent coordonner leur fonctionnement par l'intermédiaire de messages de synchronisation échangés par un mécanisme de backchannel non spécifié par la présente norme. Un seul appareil G,M,S peut être responsable de toutes les fonctions de passerelle, de gestionnaire de système et de gestionnaire de sécurité, avec un second appareil G,M,S agissant comme appareil de réserve active qui reste au repos en attendant qu'on en ait besoin. En variante, les deux appareils G,M,S peuvent diviser la charge de travail entre eux jusqu'à ce que l'un d'eux ait une défaillance.

5.3.4.4 Ajout de routeurs dorsaux

Au réseau de base représenté à la Figure 13, la Figure 14 ajoute des routeurs dorsaux (boîtes étiquetées B), qui facilitent l'extension des réseaux conformes à la présente norme, en termes de nombre d'appareils et de surface que le réseau occupe.



Légende

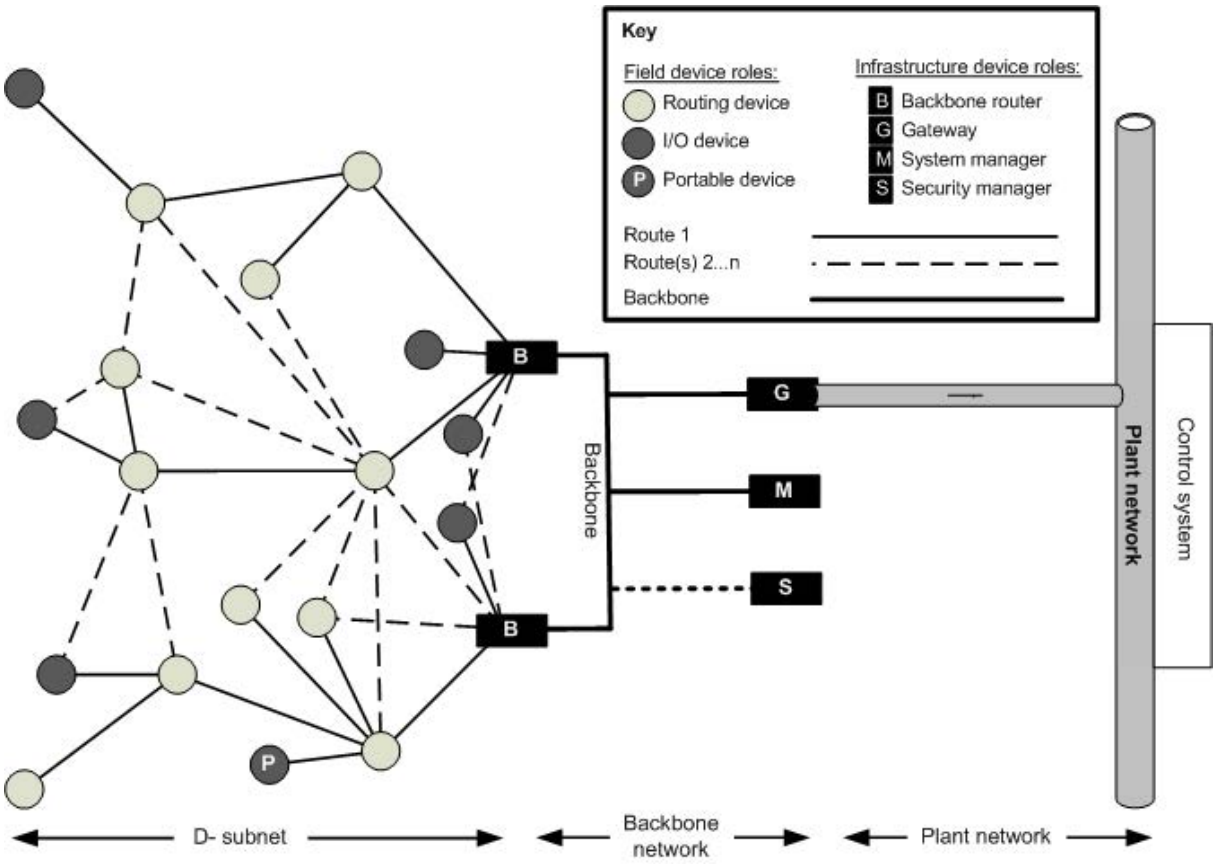
Anglais	Français
Key	Légende
Field device roles	Rôles d'appareil de terrain
Routing device	Appareil de routage
I/O device	Appareil E/S
Portable device	Appareil portatif
Infrastructure device roles	Rôles d'appareil d'infrastructure
B Backbone router	B Routeur dorsal

Anglais	Français
G Gateway	G Passerelle
M System manager	M Gestionnaire de système
S Security manager	S Gestionnaire de sécurité
Route 1	Chemin 1
Route(s) 2...n	Chemin(s) 2...n
Backbone	Dorsale
Control system	Système de commande
Plant network	Réseau d'installation
Backbone	Dorsale
Synchronization messages	Messages de synchronisation
D-subnet	Sous-réseau D
Backbone network	Réseau dorsal

Figure 14 – Réseau avec dorsale

5.3.5 Passerelle, gestionnaire de système, et gestionnaire de sécurité

Comme montrés à la Figure 15, les rôles fonctionnels remplis par l'appareil G,M,S à la Figure 10, à la Figure 11, Figure 12 et à la Figure 14 peuvent être éclatés en plusieurs appareils physiquement séparés, afin que la passerelle G, le gestionnaire de système M et le gestionnaire de sécurité S fonctionnent chacun comme un appareil distinct.



Légende

Anglais	Français
Key	Clé
Field device roles	Rôles d'appareil de terrain

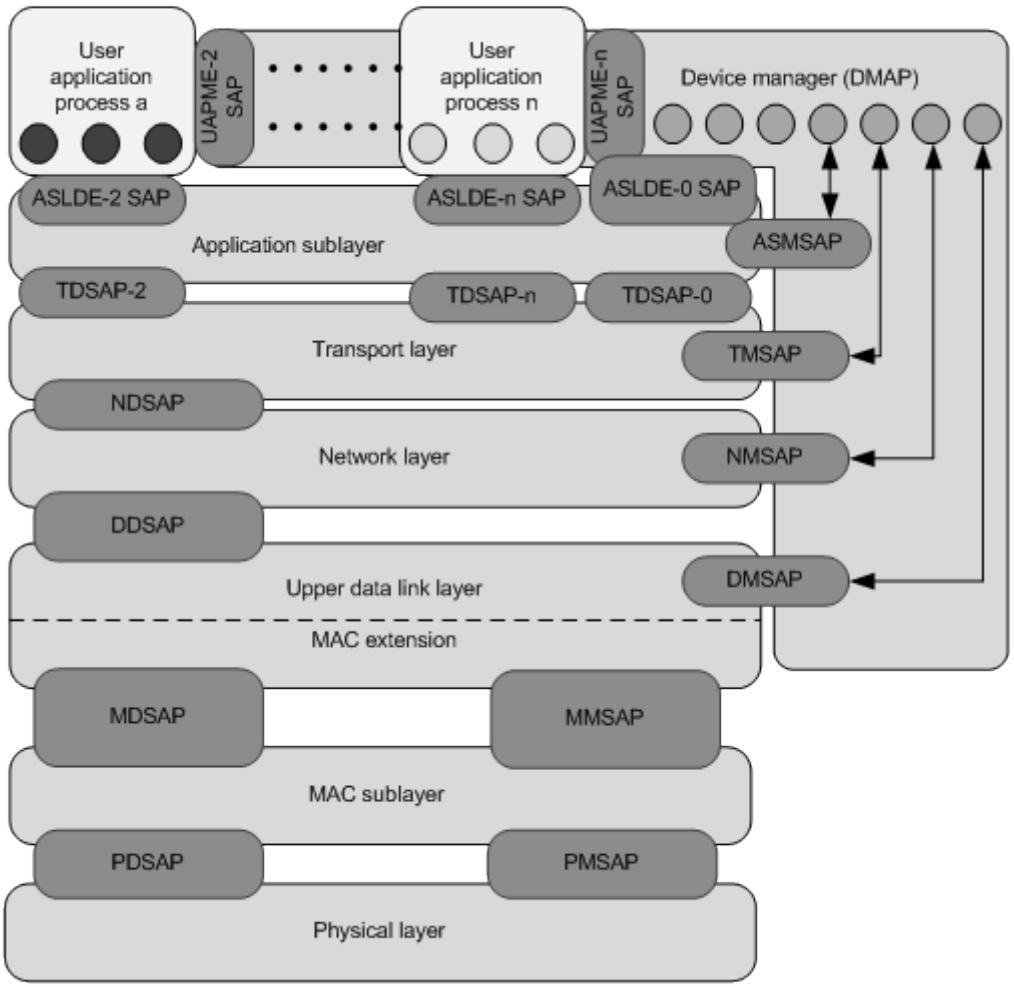
Anglais	Français
Routing device	Appareil de routage
I/O device	Appareil E/S
Portable device	Appareil portatif
Infrastructure device roles	Rôles d'appareil d'infrastructure
B Backbone router	B Routeur dorsal
G Gateway	G Passerelle
M System manager	M Gestionnaire de système
S Security manager	S Gestionnaire de sécurité
Route 1	Chemin 1
Route(s) 2…n	Chemin(s) 2…n
Backbone	Dorsale
Control system	Système de commande
Plant network	Réseau d'installation
Backbone	Dorsale
D-subnet	Sous-réseau D
Backbone network	Réseau dorsal

Figure 15 – Réseau avec dorsale – Rôles des appareils

La passerelle, le gestionnaire de système, et le gestionnaire de sécurité physiquement séparés qui sont représentés à la Figure 15 ne peuvent être mis en œuvre que dans les réseaux avec une dorsale de réseau.

5.4 Structure de suite de protocoles

Les couches de protocoles pour un appareil conforme à la présente norme sont décrites en termes du Modèle de référence de base OSI, qui est adapté tel que montré à la Figure 16. Tous les rôles et types d'appareils conformes à la présente norme peuvent être dérivés de ce modèle par extension ou restriction des éléments communs montrés à la Figure 16.



Légende

Anglais	Français
User application process a	Processus d'application utilisateur a
User application process n	Processus d'application utilisateur n
Device manager (DMAP)	Gestionnaire d'appareil (DMAP)
Application sub-layer	Sous-couche d'application
Transport layer	Couche transport
Network layer	Couche réseau
Upper data link layer	Couche liaison de données supérieure
MAC extension	Extension MAC
MAC sub-layer	Sous-couche MAC
Physical layer	Couche physique

Figure 16 – Modèle de référence utilisé par la présente norme

Comme montré à la Figure 16, chaque couche fournit un point d'accès de service (SAP). Les services d'une couche sont définis comme étant les fonctions et les capacités de la couche en question qui sont exposées par l'intermédiaire du SAP aux couches environnantes. En général, deux types de SAP sont définis: les SAP de données, qui sont utilisés pour le transfert de données opérationnelles, et les SAP de gestion, qui sont utilisés pour la gestion de couche. Les services fournis par une couche sont définis par les données circulant par les SAP de données et, dans certains cas, par les états qu'une couche fournit et les transitions d'état qui sont pilotées par l'interaction à travers ces SAP. Le gestionnaire d'appareil est l'entité au sein de chaque appareil qui accomplit la fonction de gestion; dans la plupart des cas, il est accessible par l'intermédiaire d'un SAP de gestion de couche. Le gestionnaire

d'appareil a un chemin consacré menant à plusieurs des couches de protocoles inférieures au sein d'un appareil, pour fournir la maîtrise directe en temps réel sur le fonctionnement de ces couches ainsi que l'accès direct à des informations relatives aux diagnostics et aux statuts.

Dans la Figure 16, l'appareil assure la fonctionnalité de chaque SAP de gestion utilisé par le DMAP pour chaque couche de protocoles mise en œuvre.

Etant donné que la conformité peut être évaluée uniquement sur les interfaces externes, notamment le contenu des structures de données acheminées sur ces interfaces, toutes les descriptions théoriques portant sur la manière dont les fonctionnalités spécifiques pourraient être mises en œuvre ont uniquement un but informatif, et pas normatif.

5.5 Flot de données

5.5.1 Généralités

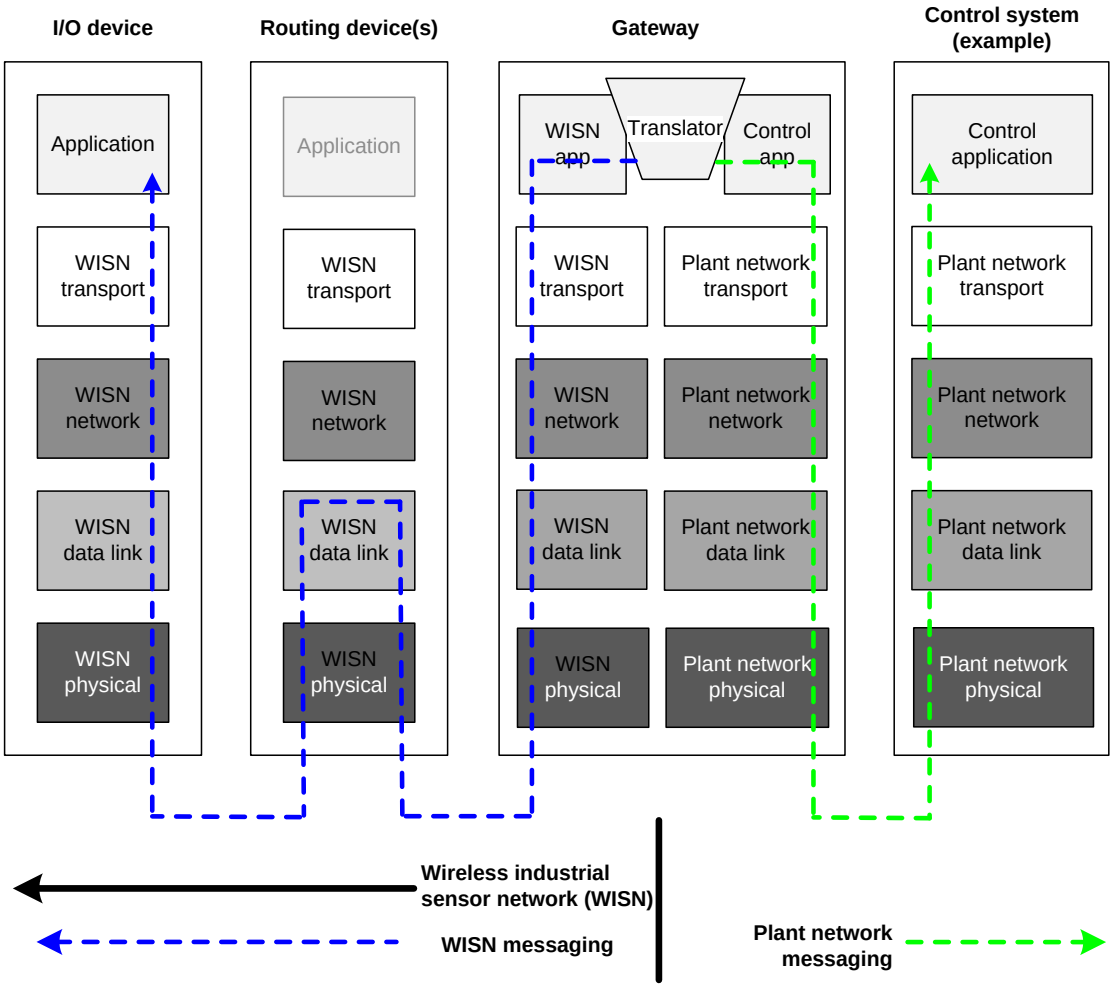
Les descriptions en 5.5 sont censées fournir des exemples de la manière dont les données peuvent circuler à travers le système. L'ensemble d'exemples n'est pas censé être exhaustif.

5.5.2 Communications natives

Un appareil communique sur le réseau en utilisant seulement des services définis par ASL tels que définis à l'Article 12; les charges utiles sont classifiées comme étant natives ou non natives. Les charges utiles natives sont définies à l'Article 12; les charges utiles non natives ne sont pas définies dans la présente norme.

5.5.3 Flot de données de base

La Figure 17 illustre le flot de données en régime établi pour un réseau de base conforme à la présente norme, tel que celui montré à la Figure 10.



Légende

Anglais	Français
I/O device	Appareil E/S
Application	Application
WISN transport	Transport WISN
WISN network	Réseau WISN
WISN data link	Liaison de données WISN
WISN physical	Physique de WISN
Routing device(s)	Appareil(s) de routage)
Gateway	Gateway
WISN app	WISN app
Translator	Convertisseur
Control app	App de commande
Plant network transport	Transport réseau d'installation
Plant network network	Réseau de réseau d'installation
Plant network data link	Liaison de données de réseau d'installation
Plant network physical	Physique de réseau d'installation
Control system (example)	Système de commande (exemple)
Control application	Application de commande
Wireless industrial sensor network (WISN)	Réseau de capteurs industriels sans fil (WISN)
WISN messaging	Messagerie WISN
Plant network messaging	Messagerie de réseau d'installation

Figure 17 – Flot de données de base

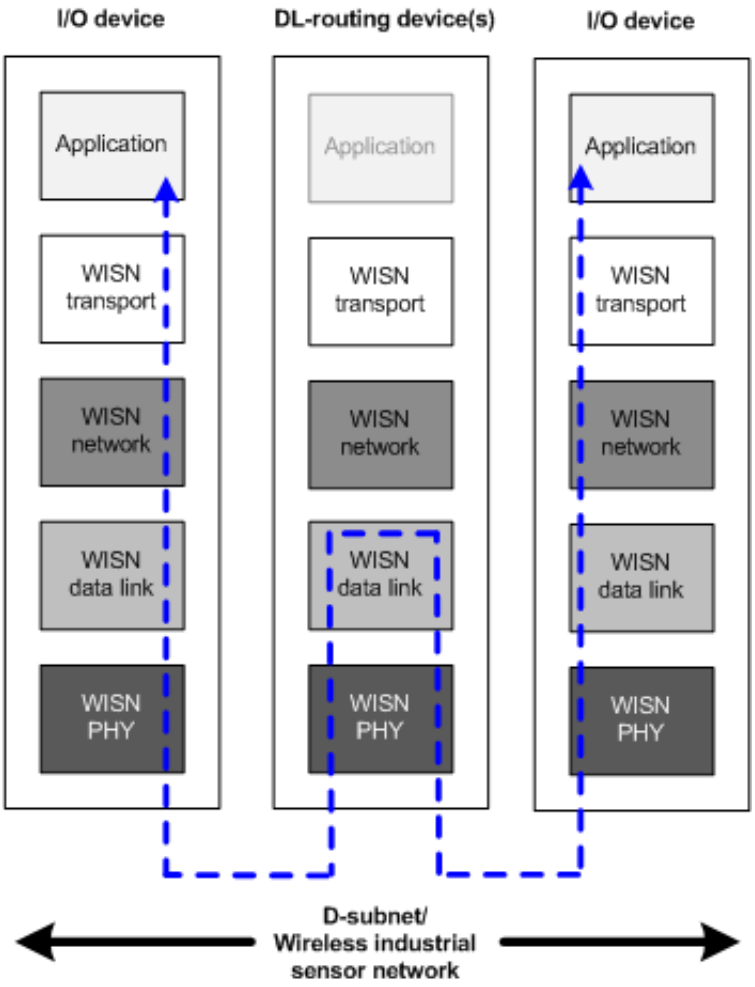
L'appareil E/S est un appareil capteur ou actionneur au sein du sous-réseau D qui contient des couches physiques, liaison de données, réseau et transport telles que définies par la présente norme et exécute une application qui traite la fonction de capteur ou d'actionneur.

Le routeur achemine des messages pour le compte de l'appareil E/S. Le routage au sein du sous-réseau D est entièrement accompli au sein de la DL, et pas au sein de la NL (voir Article 9). Dans un réseau du monde réel, il y aura un routeur pour chaque saut supplémentaire entre l'appareil et le routeur relié au WISN ou le routeur dorsal.

La passerelle traduit les messages entre le sous-réseau D et le réseau d'installation. L'application tournant sur la passerelle est constituée d'une composante qui communique avec l'ALE de l'appareil E/S, plus une composante qui communique avec une ALE au sein du système de contrôle, plus toutes les éventuelles composantes qui facilitent la traduction entre les deux, telles qu'une mémoire cache.

5.5.4 Flot de données entre appareils E/S

La Figure 18 montre le flot de données pour la communication entre les appareils E/S au sein d'un sous-réseau D. Le routage au sein du sous-réseau D est entièrement accompli au sein de la DL, et pas au sein de la NL.



Légende

Anglais	Français
I/O device	Appareil E/S
Application	Application
WISN transport	Transport WISN

Anglais	Français
WISN network	Réseau WISN
WISN data link	Liaison de données WISN
WISN PHY	PHY WISN
DL-routing device(s)	Appareil(s) de routage DL
D-subnet/Wireless industrial sensor network	Sous-réseau D/Réseau de capteurs industriel sans fil

Figure 18 – Flot de données entre appareils E/S

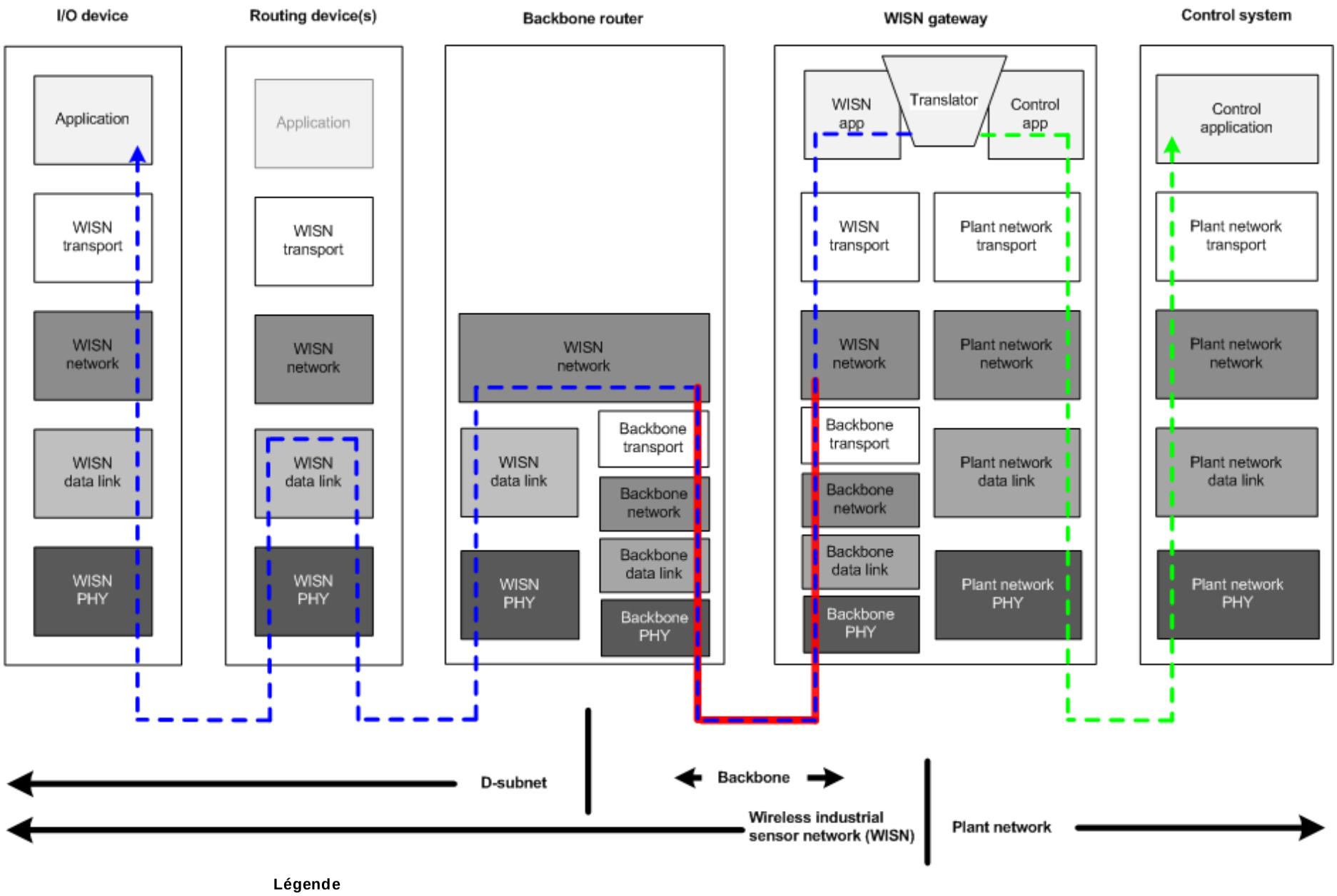
5.5.5 Flot de données avec un appareil E/S hérité

La Figure 19 montre un appareil E/S hérité qui est intégré dans un sous-réseau D par l'intermédiaire d'un adaptateur d'appareil hérité. Un adaptateur est une spécialisation d'un appareil qui convertit en interne le protocole de l'appareil ou des appareils hérités connectés dans celui d'un appareil réseau conforme à la présente norme.

Légende

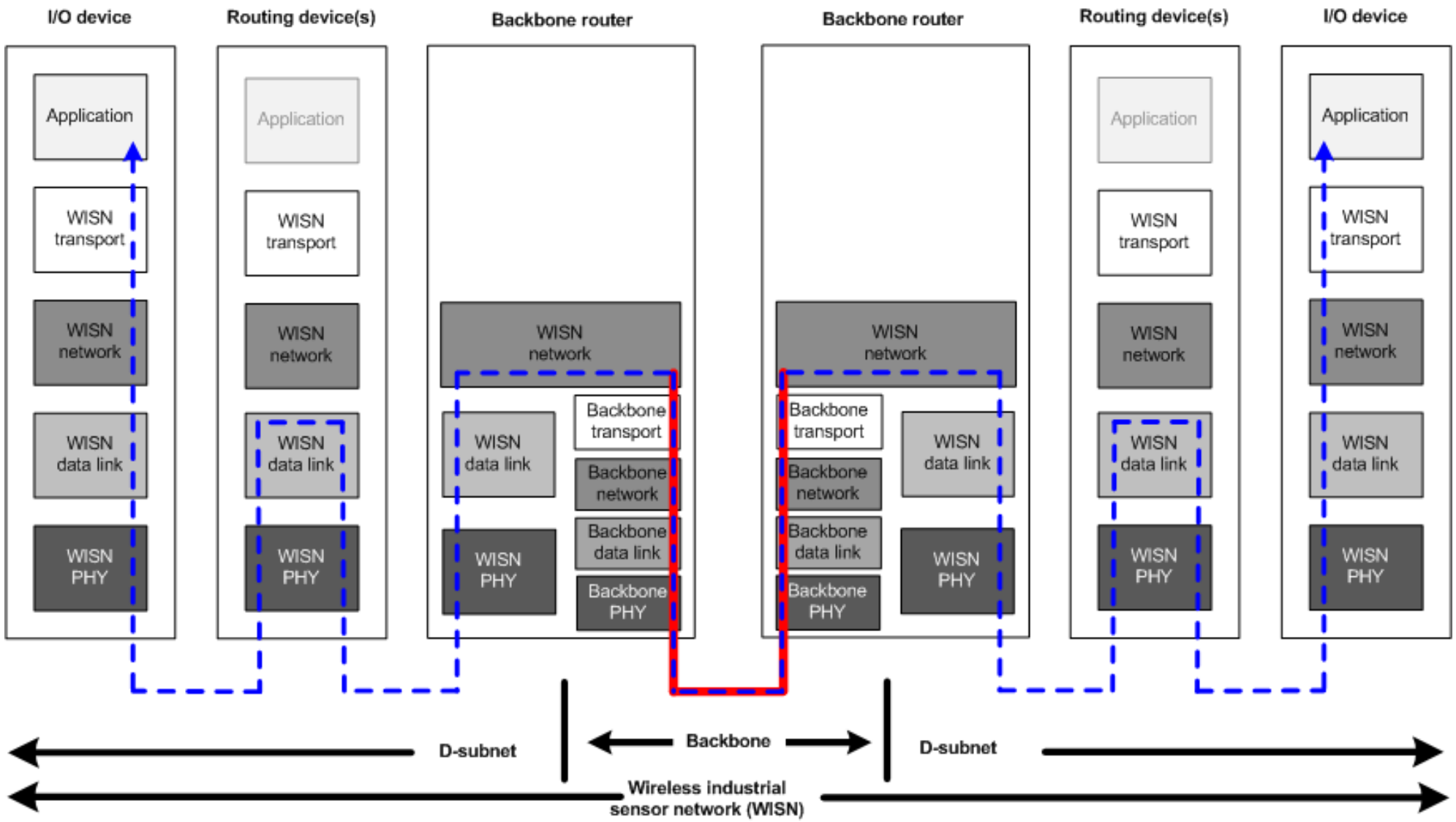
Anglais	Français
Legacy I/O device	Appareil E/S hérité
Application	Application
Legacy transport (if present)	Transport hérité (si présent)
Legacy network (if present)	Réseau héritée (si présent)
Legacy data link	Liaison de données héritée
Legacy PHY	PHY héritée
Legacy device WISN adapter	Adaptateur WISN d'appareil hérité
Legacy app	App héritée
Translator	Convertisseur
WISN app	App WISN
WISN transport	Transport WISN
WISN network	Réseau WISN
WISN data link	Liaison de données WISN
WISN PHY	PHY WISN
Routing device(s)	Appareil(s) de routage
Application	Application
WISN gateway	Passerelle WISN
Control app	App de commande
Plant network transport	Transport de réseau d'installation
Plant network network	Réseau de réseau d'installation
Plant network data link	Liaison de données de réseau d'installation
Plant network PHY	PHY de réseau d'installation
Control system	Systèmes de commande
Control application	Application de commande
Legacy network	Réseau existant
D-subnet/Wireless industrial sensor network (WISN)	Sous-réseau D/Réseau de capteurs industriel sans fil (WISN)
Plant network	Réseau d'installation

Figure 19 – Flot de données avec appareil E/S hérité



Anglais	Français
I/O device	Appareil E/S
Application	Application
WISN transport	Transport WISN
WISN network	Réseau WISN
WISN data link	Liaison de données WISN
WISN PHY	PHY WISN
Routing device(s)	Appareil(s) de routage
Backbone router	Routeur dorsal
Backbone transport	Transport de dorsale
Backbone network	Réseau dorsal
Backbone data link	Liaison de données de dorsale
Backbone PHY	PHY de dorsale
WISN gateway	Passerelle WISN
WISN app	App WISN
Translator	Convertisseur
Control app	App de commande
Plant network transport	Transport de réseau d'installation
Plant network network	Réseau de réseau d'installation
Plant network data link	Liaison de données de réseau d'installation
Plant network PHY	PHY de réseau d'installation
Control system	Système de commande
Control application	Application de commande
D-subnet	Sous-réseau D
Backbone	Dorsale
Wireless industrial sensor network (WISN)	Réseau de capteurs industriels sans fil
Plant network	Réseau d'installation

Figure 20 – Flot de données avec appareil résidant sur la dorsale



Légende

Anglais	Français
I/O device	Appareil E/S
Application	Application
WISN transport	Transport WISN
WISN network	Réseau WISN
WISN data link	Liaison de données WISN
WISN PHY	PHY WISN
Routing device(s)	Appareil(s) de routage
Backbone router	Routeur dorsal
Backbone transport	Transport de dorsale
Backbone network	Réseau dorsal
Backbone data link	Liaison de données de dorsale
Backbone PHY	PHY de dorsale
D-subnet	Sous-réseau D
Backbone	Dorsale
Wireless industrial sensor network (WISN)	Réseau de capteurs industriels sans fil

Figure 21 – Flot de données entre appareils E/S par un sous-réseau dorsal

5.5.6 Flot de donnée avec dorsale

La Figure 20 introduit un routeur dorsal dans le flot de données.

Le routeur dorsal encapsule des NPDU et les relaie à travers les couches physiques, liaison de données, réseau et transport de la dorsale. La passerelle utilise les mêmes couches de dorsale pour récupérer les NPDU. Bien que cela ne soit pas montré à la Figure 20, la passerelle peut inclure à la fois une PhLE et une DLE telles que définies par la présente norme, permettant à la passerelle de traiter des messages directement issus d'un sous-réseau D, en plus des messages relayés par une interface dorsale.

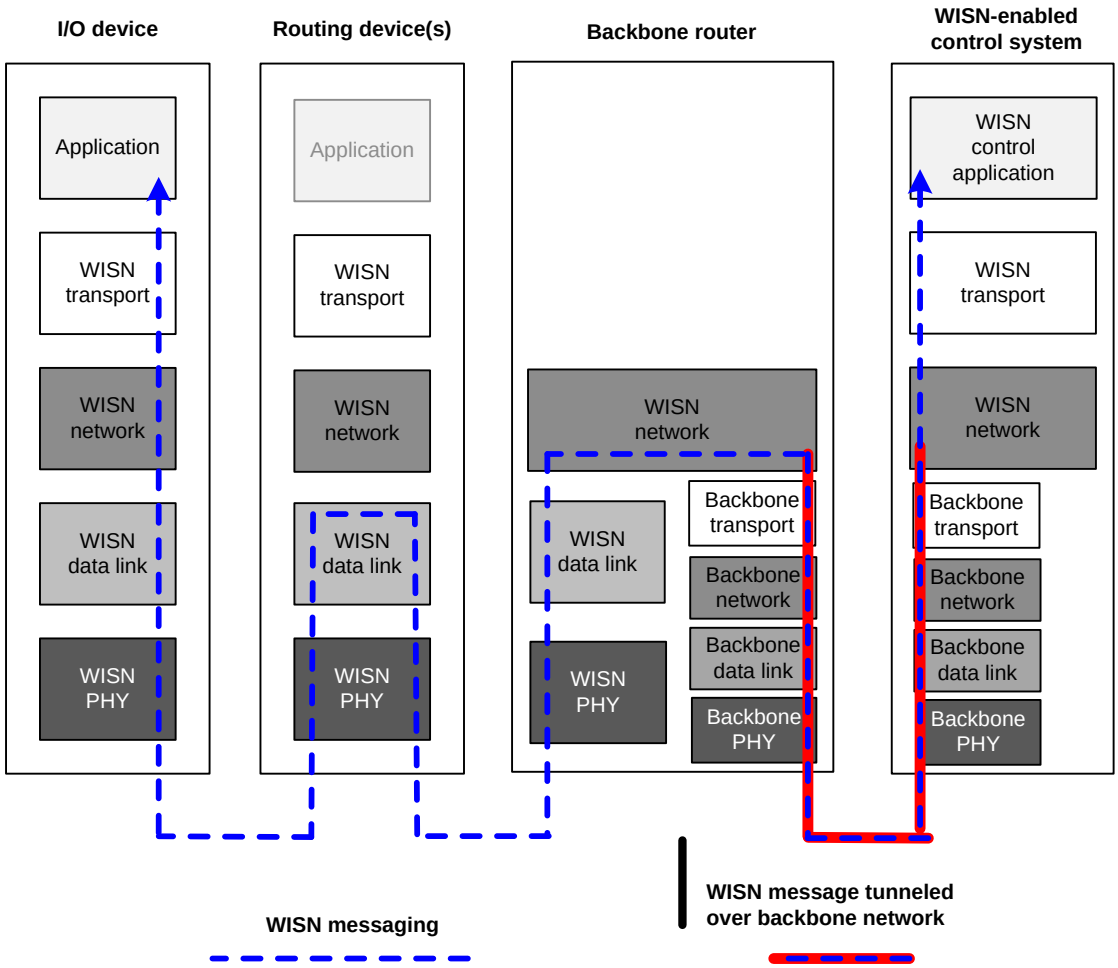
5.5.7 Flot de données entre appareils E/S par une dorsale

La Figure 21 montre comment un réseau dorsal traite le transfert de messages conforme à la présente norme lorsqu'un appareil E/S communique directement avec un autre appareil E/S dans un sous-réseau D différent.

5.5.8 Flot de données vers un système de contrôle ou appareil compatible à la norme

Un système de commande compatible avec la norme est un système de commande qui comprend la messagerie définie par la présente norme et n'a pas besoin d'une passerelle pour accomplir la conversion de protocole. La Figure 22 montre le flux de données à destination d'un système de commande compatible avec la norme.

NOTE La conversion de protocole générique est traitée à l'Annexe O.



Légende

Anglais	Français
I/O device	Appareil E/S
Application	Application
WISN transport	Transport WISN
WISN network	Réseau WISN
WISN data link	Liaison de données WISN
WISN PHY	PHY de WISN
Routing device(s)	Appareil(s) de routage
Backbone router	Routeur dorsal
Backbone transport	Transport de dorsale
Backbone network	Réseau dorsal
Backbone data link	Liaison de données de dorsale
Backbone PHY	PHY de dorsale
WISN-enabled control system	Système de commande adapté à WISN
WISN control application	Application de commande WISN
WISN message tunneled over backbone network	Message WISN tunnellisé sur réseau dorsal
WISN messaging	Message WISN

Figure 22 – Flux de données à destination d'un système de commande compatible avec la norme

En général, pour qu'un appareil soit compatible avec la norme, il a seulement besoin de prendre en charge l'interface d'application définie par la présente norme et de mettre en œuvre les couches application, transport et réseau définies par la présente norme. Cela permet à deux appareils compatibles avec la norme de communiquer par l'intermédiaire d'un réseau d'installation sans utiliser ou requérir de sous-couches.

5.6 Référence temporelle

5.6.1 Généralités

Le temps selon la présente norme est basé sur le temps atomique international (TAI) comme référence temporelle; voir 6.3.10. Le temps selon la présente norme est communiqué sous la forme de secondes écoulées depuis l'instant TAI de 00:00 le 1er janvier 1958 (c'est-à-dire 1958/01/01 00:00).

Il n'est pas possible ou même souhaitable que chaque réseau suive précisément une horloge atomique. Au contraire, chaque réseau doit avoir un sens du temps qui:

- est à croissance monotone à une fréquence qui coïncide étroitement avec le temps réel;
- ne dépasse pas une erreur de plus de 1 s par rapport à la source du temps système; et
- est livré aux diverses couches dans des appareils de terrain en unités de TAI cohérentes.

Il existe des modes de communication tels que définis à l'Article 9 qui exigent une exactitude d'horloge meilleure que 1 s par rapport à la source de temps système.

Pour le fonctionnement du protocole, la séquence de rapports relatifs aux événements, et autres besoins, il est habituellement nécessaire de diviser le temps en incréments de moins d'une seconde. Par exemple, les incréments peuvent être représentés de plusieurs façons:

Tops d'horloge de WISN: Deux octets pour marquer le temps en incréments de 2^{-15} s (32 768 Hz, ou ~30,52 μs par top).

Précision en microsecondes: Trois octets pour marquer le temps en incréments de 2^{-20} s (~0,95 μs par incrément).

Précision en nanosecondes: Quatre octets pour marquer le temps en incréments de 2^{-30} s (~0,93 ns par incrément).

Les appareils ayant besoin de convertir le temps TAI vers le format hh:mm:ss, tel que sur un écran utilisateur, peuvent rendre compte d'un réajustement de secondes intercalaires accumulées en temps universel coordonné (TUC). Ce réajustement est disponible pour les appareils de terrain auprès du gestionnaire de système. Si le réajustement TUC est utilisé par un appareil de terrain, il convient qu'il rafraîchisse le réajustement au début de chaque mois.

NOTE Une liste de tels réajustements TUC est maintenue à l'adresse ftp://maia.usno.navy.mil/ser7/tai-utc.dat.

Les demandes de mise à jour TUC faites simultanément par un grand nombre d'appareils peuvent causer un orage d'activité dans la DL. Il convient de prendre cet élément en compte dans la conception du DMAP; son évitement n'est pas couverte par la courante spécification de la DL.

Tous les appareils dans un réseau partagent la référence temporelle TAI avec des degrés variables d'exactitude. Chaque appareil au sein d'un réseau doit maintenir le temps avec une exactitude à 1 s près.

Le gestionnaire de système dirige des appareils sur le système vers un appareil mettant en œuvre le rôle de la source de temps système. Dans la plupart des cas, cet appareil remplira également le rôle de gestionnaire de système. Cependant, la responsabilité de la source de temps peut être redirigée vers n'importe quel appareil ayant une source de temps plus capable.

La passerelle doit être responsable de convertir entre le temps TAI nominal du réseau et une référence temporelle externe autre que TAI, s'il en est utilisée une.

Pour plus d'informations sur les exigences relatives à la source de temps, voir 6.3.10.

5.6.2 Synchronisation du temps

Pour propager le temps d'hôte, une passerelle peut périodiquement synchroniser le sens du temps dans un sous-réseau D rattaché à une source de temps externe en demandant des changements de temps par l'intermédiaire des DLMO.

Le WISN fournit une synchronisation pour des applications afin que, au niveau de l'appareil, elles puissent utiliser le temps pour coordonner des activités ou horodater des données, une activité qui pourrait améliorer l'utilisation de l'énergie et augmenter la fiabilité. Le temps système doit être disponible auprès d'au moins un appareil (une source de temps système) sur chaque WISN. Voir Article 9.

5.7 Mises à niveau de firmware

Le système global, et chaque appareil sur le WISN, doivent fournir la capacité de mettre à niveau le firmware d'appareil qui met en œuvre la présente norme par l'intermédiaire du réseau sans fil (voir 6.3.6). Le système doit prendre en charge un mécanisme commun, tel qu'un déclencheur basé sur le temps, pour informer tous les appareils de commuter simultanément vers un nouveau firmware; ce mécanisme peut être employé pour réduire au maximum le nombre d'appareils qui sont laissés bloqués avec une suite incompatible de protocoles réseau. Les mécanismes de sécurité intégrés dans la présente norme sont utilisés au cours de la mise à niveau d'un firmware.

Chaque version du protocole doit prendre en charge les versions antérieures dans toute la mesure nécessaire pour permettre la mise à niveau du firmware via le réseau sans fil.

5.8 Dorsales sans fil et autres infrastructures

Les appareils conformes à la présente norme sont des appareils gérés. Tous les appareils conformes à la présente norme doivent mettre en œuvre les interfaces de gestion d'appareil à chaque couche, mais ils peuvent ne mettre en œuvre que la fonctionnalité de leurs couches fonctionnelles exigées.

Le système prend en charge les réseaux dorsaux tant câblés que sans fil par l'utilisation des routeurs dorsaux. Le fonctionnement des réseaux dorsaux n'est pas traité par la présente norme.

Plus d'informations relatives aux réseaux dorsaux et à leurs caractéristiques impliquées peuvent être consultées à l'Annexe E.

6 Rôle de gestion de système

6.1 Généralités

6.1.1 Vue d'ensemble

Le rôle de gestion de système prend en charge la gestion de réseau du réseau comme un tout, ainsi que la gestion d'appareil des appareils fonctionnant au sein du réseau. La gestion de réseau inclut la gestion des diverses ressources de communications à travers le réseau et à travers toutes les couches de protocoles de l'architecture. La gestion d'appareil prend en charge la gestion localisée des ressources de communications, et potentiellement d'autres ressources, d'un appareil.

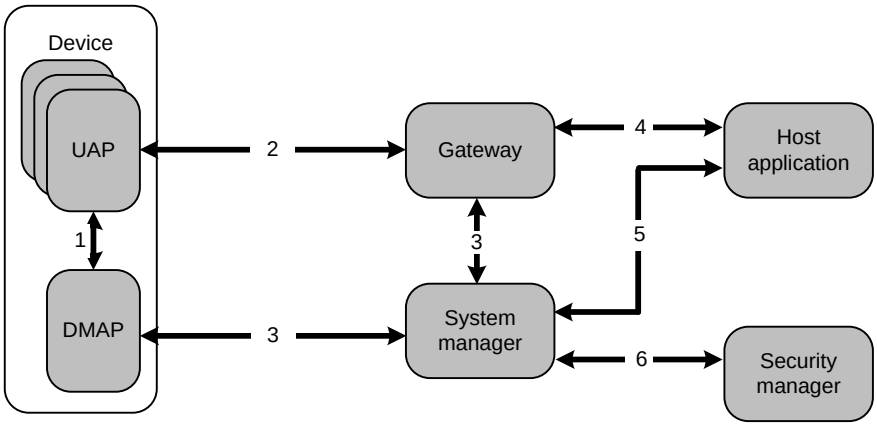
Les fonctions de gestion décrites par la présente norme prennent en charge:

- le rattachement au réseau et départ du réseau;
- les rapports relatifs aux défauts qui se produisent au sein du réseau;
- la configuration de communication;
- la configuration de la distribution d'horloges et le réglage du temps système;
- la surveillance d'appareil;
- la surveillance et optimisation de performances; et
- la configuration et surveillance de la sécurité.

6.1.2 Composantes et architecture

Les composantes primaires du service de gestion incluent un processus d'application de gestion d'appareil (DMAP) qui réside sur chaque appareil conforme à la présente norme, ainsi qu'un processus d'application de gestion de système (SMAP) qui doit résider sur un appareil qui met en œuvre le rôle de gestionnaire de système. Les rôles sont décrits en 5.2.6.5. Le DMAP est un type spécial de processus d'application utilisateur (UAP) qui est consacré à gérer l'appareil et ses services de communications, décrits en 12.4.3 et 12.4.4. Le DMAP et le gestionnaire de système doivent être capables de communiquer l'un avec l'autre sur le réseau en utilisant les services de sous-couche d'application définis par la présente norme et doivent fournir ensemble un moyen d'accéder aux informations de gestion à distance et gérer le système et ses appareils. La gestion de système est accomplie par l'intermédiaire d'une messagerie entre appareils (inter-appareil), alors que la gestion d'appareil est accomplie par des communications locales internes à un appareil (intra-appareil).

L'architecture de gestion selon la présente norme est montrée à la Figure 23.



Légende

Anglais	Français
Device	Appareil
Gateway	Passerelle
System manager	Gestionnaire de système
Host application	Application d'hôte
Security manager	Gestionnaire de sécurité

Figure 23 – Architecture de gestion

Les appareils conformes à la présente norme doivent être gérés par l'intermédiaire de deux classes distinctes de processus applicatifs, à savoir les UAP et le DMAP. Les UAP sont configurés et surveillés par des applications d'hôtes, telles que les systèmes de gestion automatisés, ou par des applications de proxys d'hôtes dans les passerelles. Le DMAP dans l'appareil doit être géré par le gestionnaire de système.

La Figure 23 montre les relations des modèles de gestion selon la présente norme. Pour les chemins montrés à la Figure 23, la présente norme fournit une description normative des protocoles de communication pour les chemins 2 et 3. Les protocoles de communication pour les chemins 1, 4, 5 et 6 sont des exemples informatifs des mises en œuvre dans la présente norme.

La présente norme définit les protocoles de communication de gestion de système qui doivent être utilisés pour commander et surveiller les DMAP dans le réseau et les chemins de communication correspondants. Dans ce cas, ces communications parcourent le chemin 3 à la Figure 23.

Le gestionnaire de système inclut les chemins de communication à l'extérieur du réseau conforme à la présente norme qui permettent à d'autres appareils d'interagir avec lui. A la Figure 23, le chemin 5 montre une connexion entre le gestionnaire de système et l'application d'hôte. Ce chemin permet à l'application d'hôte de récupérer le statut de réseau et de demander des services de réseau. Le gestionnaire de système communique également avec le gestionnaire de sécurité sur le chemin 6 pour configurer la sécurité dans le réseau et pour rapporter le statut.

Les applications d'utilisateur sur des appareils conformes à la présente norme communiquent avec des passerelles et des applications d'hôte en utilisant les protocoles normalisés montrés sur le chemin 2 à la Figure 23. Cela est décrit à l'Article 12 et Annexe U.

L'application d'hôte basée dans l'installation communique avec la passerelle en utilisant des protocoles d'installation qui parcourent le chemin 4. Le gestionnaire de système ne communique pas directement avec les UAP. Il y a un chemin de communication intra-appareil qui permet aux processus DMAP et UAP d'interagir par l'intermédiaire du chemin de

communication intra-appareil 1 à la Figure 23 entre le gestionnaire de système et une passerelle. Cela est accompli sur une interface virtuelle 1 à la Figure 23, UAPME-SAP, utilisant l'objet de gestion UAP (UAPMO) qui est décrit en 12.15.2.2.

6.1.3 Fonctions de gestion

Chaque réseau qui est conforme à la présente norme doit inclure au moins un rôle de gestion de système et un rôle de gestion de sécurité. Ces rôles doivent être accessibles à tous les appareils conformes à la norme sur ce réseau.

La gestion de système est un rôle spécialisé qui régit le réseau, le fonctionnement des appareils sur le réseau, et les communications du réseau. Les fonctions définies dans ce rôle sont accomplies par le gestionnaire de système, fournissant la commande politique de la configuration des communications en temps d'exécution. Le gestionnaire de système accomplit une surveillance et des rapports sur la configuration des communications, les performances, les conditions de défauts, et le statut opérationnel. Cela est décrit en 6.3.7. Le gestionnaire de système fournit également des services relatifs au temps. Certaines fonctions de gestion de système peuvent être complètement automatisées, alors que d'autres peuvent être assistées par l'homme.

Le gestionnaire de système prend en charge la configuration du réseau conforme à la présente norme, y compris les attributs de la suite de protocoles de la DL vers l'AL pour des applications de gestion de système. Il gère l'établissement, la modification et la résiliation de contrats qui sont utilisés par les appareils conformes à la présente norme pour communiquer les uns avec les autres. Les fonctions du gestionnaire de système n'incluent pas la commande, la configuration, et la surveillance des UAP sur l'appareil. Ces fonctions de gestion sont commandées par des applications d'hôtes sur des réseaux d'installation ou dans des outils de maintenance tenus à la main.

La gestion de sécurité du système est une fonction spécialisée qui est réalisée en une entité et qui travaille conjointement avec la fonction de gestion de système pour permettre l'exploitation sécurisée du système. Cette fonction est accomplie par le gestionnaire de sécurité. Certaines fonctions de gestion de sécurité peuvent être complètement automatisées, alors que d'autres peuvent être assistées par l'homme.

Chaque appareil conforme à la présente norme doit contenir un DMAP. Le DMAP inclut une fonction locale de gestion de degré de sécurité d'appareil. Le DMAP coopère avec le gestionnaire de système et le gestionnaire de sécurité pour permettre l'utilisation des ressources du système par l'appareil et la gestion sécurisée des ressources d'un appareil. Par exemple, le DMAP peut demander à rejoindre le réseau, demander de la largeur de bande de communication, demander une configuration de communication, et rapporter sa santé. Le gestionnaire de système et le gestionnaire de sécurité autorisent l'appareil à rejoindre le réseau, allouent la largeur de bande de communication, configurent l'appareil, et rassemblent les rapports de santé. Ces rapports de santé sont stockés dans le gestionnaire de système et sont utilisés pour prendre des décisions de configuration de communication.

Afin de compartimenter les fonctions de sécurité, l'architecture de gestion définie par la présente norme prend en charge des fonctions séparables de gestion de système et de gestion de sécurité tant au niveau du système qu'au niveau de l'appareil. Ainsi, le gestionnaire de sécurité est logiquement séparable du gestionnaire de système. Plus de détails relatifs au gestionnaire de sécurité sont fournis à l'Article 7.

NOTE Les fonctions de gestion de système et de gestion de sécurité sont souvent incluses au sein d'une seule et même entité physique.

6.2 DMAP

6.2.1 Généralités

Le DMAP est un type spécial de processus d'application consacré à gérer l'appareil conforme à la présente norme et ses services de communications. Un DMAP réside sur chaque appareil conforme à la présente norme.

6.2.2 Architecture de la gestion d'appareil

Telle que montrée à Figure 16, la structure de suite de protocoles d'un appareil inclut les couches de protocoles de mise en réseau et les UAP.

Le DMAP est montré en relation aux autres composants de la suite de protocoles sur la droite de la structure de la suite de protocoles, y compris des flèches décrivant l'accès aux SAP de gestion pour plusieurs des couches de protocoles. Les composantes dans le DMAP sont modélisées sous la forme d'objets, appelés objets de gestion, qui ont des caractéristiques qui sont accessibles sur le réseau. Le DMAP, comme tous les processus d'application, est capable d'utiliser la sous-couche d'application pour communiquer. Le DMAP doit utiliser le SAP de sous-couche d'application ASLDE-0 SAP pour des communications de données normales. Ce SAP de sous-couche d'application doit correspondre au SAP de TL TDSAP-0 qui doit correspondre au numéro de port 0xF0B0. La sous-couche d'application fournit des services de communication pour permettre aux objets au sein du DMAP d'interagir avec le gestionnaire de système du réseau. Ces services de communication sont décrits en 12.17.

6.2.3 Définition des objets de gestion

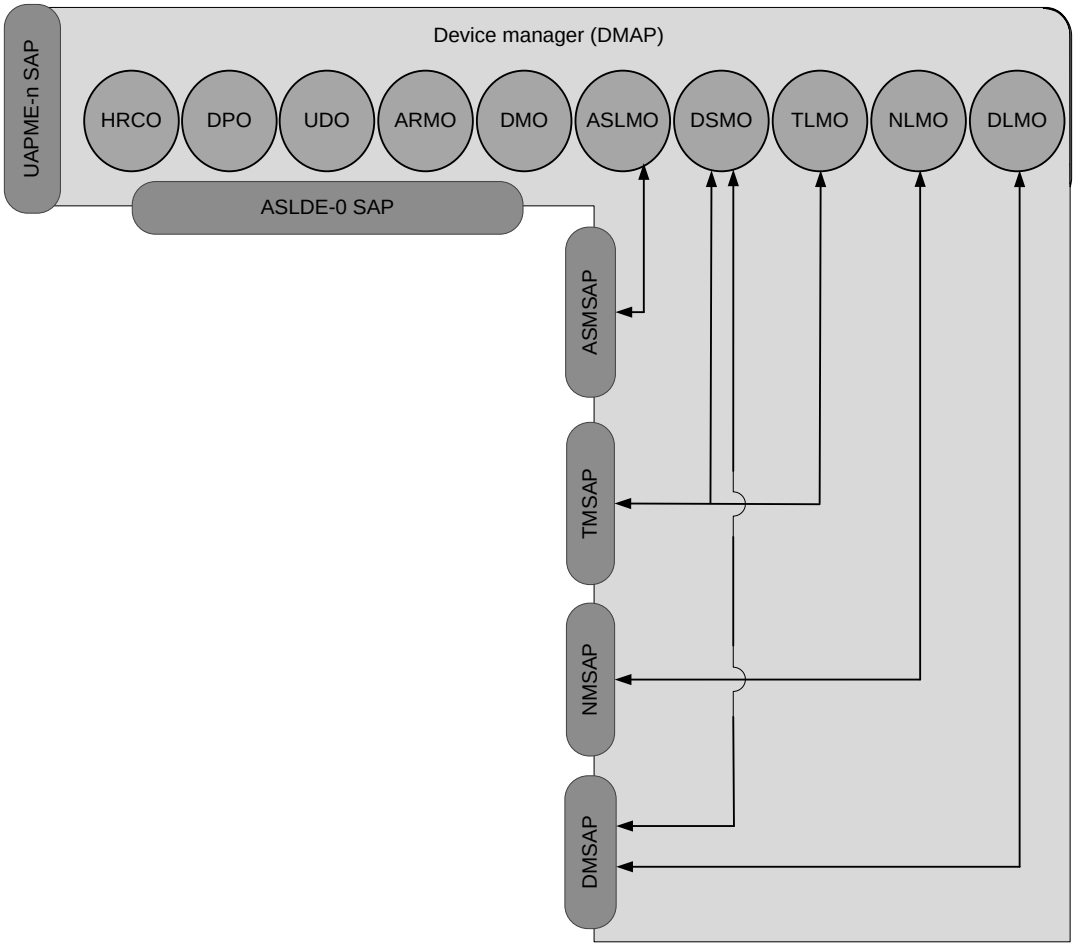
Les objets définis dans le DMAP suivent la spécification qui est utilisée pour définir les objets UAP. Les modèles pour définir les types d'objet, les attributs d'objet, les méthodes d'objet et les alertes d'objet sont spécifiés à l'Annexe I.

Les objets de gestion sont extensibles par les fabricants d'appareils et les développeurs d'appareils/suites de protocoles de réseau. Cela est décrit en 12.5. L'espace d'identification d'attributs et de méthodes est mis de côté pour les objets spécifiques à un appareil définis par un fabricant. Le gestionnaire de système ne doit pas être tenu de mettre en œuvre la prise en charge pour des extensions propriétaires pour que l'appareil en question interfonctionne et accomplisse sa fonction principale.

6.2.4 Objets de gestion dans le DMAP

Le DMAP doit contenir un certain nombre d'objets de gestion qui prennent en charge les opérations de gestion d'appareil. Ces objets doivent accomplir collectivement deux types de fonctions de gestion d'appareil. En premier lieu, ces objets doivent gérer l'appareil localement en manipulant des attributs et en invoquant des méthodes sur les SAP de gestion de couche. En second lieu, les objets de gestion doivent être accessibles à distance en utilisant les services d'ASL afin qu'un gestionnaire de système puisse manipuler des attributs et invoquer des méthodes sur les objets de gestion d'appareil ou capturer des alertes issues des objets. Ces objets sont conceptuels en ce qu'il n'existe pas d'exigences de mises en œuvre orientées objet dans l'appareil, excepté que le comportement visible extérieurement en termes de messagerie d'ASL par liaison radio doit être compatible avec le modèle de communications d'objets ayant les attributs, les méthodes, et les alertes spécifiés.

Comme montré à la Figure 24, le DMAP doit inclure un ensemble d'objets de gestion de couche, un objet de gestion d'appareil, un objet de gestion de sécurité d'appareil, un objet de gestion de rapports d'alertes, un objet de téléchargement montant/téléchargement descendant, et d'autres objets de gestion.



Légende

Anglais	Français
Device manager (DMAP)	Gestionnaire d'appareil (DMAP)

Figure 24 – DMAP

Les objets de gestion normalisés définis dans la présente norme sont donnés dans le Tableau 1.

Tableau 1 – Types d'objets de gestion normalisés dans le DMAP

Nom du type d'objet normalisé	Identificateur du type d'objet normalisé	Identificateur de l'objet normalisé	Description de l'objet
Device management object (DMO)	127	1	Cet objet facilite la gestion des fonctions générales à l'échelle de l'appareil; voir 6.2.7.1
Alert reporting management object (ARMO)	126	2	Cet objet facilite la gestion des fonctions de rapports d'alertes de l'appareil; voir 6.2.7.2
Device security management object (DSMO)	125	3	Cet objet facilite la gestion des fonctions de sécurité de l'appareil; voir 6.2.7.5
DL management object (DLMO)	124	4	Cet objet facilite la gestion de la DLE de l'appareil, voir 6.2.8.2.2
NL management object (NLMO)	123	5	Cet objet facilite la gestion de la NLE de l'appareil, voir 6.2.8.2.2.5
TL management object (TLMO)	122	6	Cet objet facilite la gestion de la TLE de l'appareil, voir 6.2.8.2.2.6
Application sublayer management object (ASLMO)	121	7	Cet objet facilite la gestion de l'ALE de l'appareil, voir 6.2.8.2.2.8

Nom du type d'objet normalisé	Identificateur du type d'objet normalisé	Identificateur de l'objet normalisé	Description de l'objet
Upload/download object (UDO)	3	8	Cet objet facilite la gestion des fonctions de téléchargement montant/descendant de l'appareil; voir 0
Device provisioning object (DPO)	120	9	Cet objet facilite la configuration de l'appareil avant qu'il ne rejoigne un sous-réseau D; voir 6.2.7.6
Health reports concentrator object (HRCO)	128	10	Cet objet facilite la publication périodique des rapports de santé d'appareil au gestionnaire de système; voir 6.2.7.7
Réservé pour les éditions futures de la présente norme	119..114	-	-

6.2.5 Services de communication fournis aux objets de gestion d'appareil

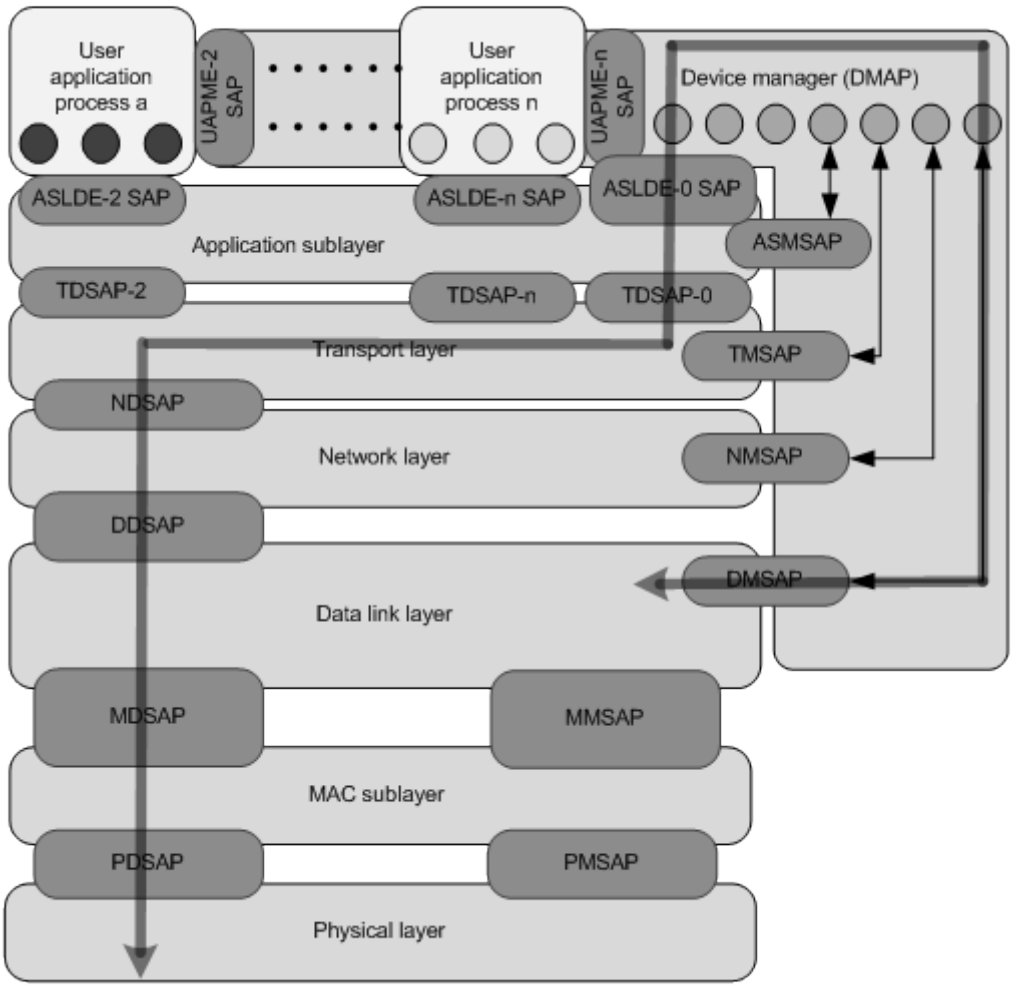
Les services de niveau application fournis aux objets de DMAP sont les mêmes que ceux fournis par la sous-couche d'application aux objets d'UAP. Ces services incluent le client/serveur (C/S), éditeur/abonné (P/S), source/puits (S/S), et rapports d'alertes (AR). Les détails de ces services, fournis par la sous-couche d'application, sont donnés en 12.17.

Comme montré à la Figure 25, TDSAP-0, qui correspond au numéro de port 0xF0B0, doit être utilisé pour accéder aux objets de gestion dans le DMAP, qui accèdent à leur tour aux attributs de gestion de couche par l'intermédiaire du SAP de gestion de couche.

L'accès aux objets de gestion d'appareil est protégé par les mécanismes de sécurité de TL décrits en 11.3.

L'accès aux objets du DMAP est limité au SMAP avec les exceptions suivantes:

- L'objet ARMO peut également être accessible par les maîtres d'alertes qui reçoivent des alertes provenant de l'appareil. Voir 6.2.7.2.3.
- Un appareil candidat au rattachement est autorisé à accéder aux méthodes de l'objet de gestion d'appareil utilisées au cours du processus de rattachement à un sous-réseau. Voir 6.3.9.2.2.



Légende

Anglais	Français
User application process a	Processus d'application utilisateur a
User application process n	Processus d'application utilisateur n
Application sub-layer	Sous-couche d'application
Transport layer	Couche transport
Network layer	Couche réseau
Data link layer	Couche liaison de données
MAC sub-layer	Sous-couche MAC
Physical layer	Couche physique
Device manager (DMAP)	Gestionnaire d'appareil (DMAP)

Figure 25 – Exemple de flux de SAP de gestion à travers une suite de protocoles normalisée

Les interactions de C/S (y compris la lecture et l'écriture d'attributs, l'exécution de méthodes, le rattachement, la demande et la fourniture de contrats) sont les principaux outils utilisés pour la gestion de système. En outre, le DMAP peut utiliser les services de rapports d'alertes de l'ASL pour faire un rapport au gestionnaire de système lorsque certaines conditions relatives à la gestion sont détectées. Les concepteurs des objets de gestion peuvent utiliser des alertes à divers niveaux de priorité pour aider à accomplir des fonctions de gestion de système et d'appareil.

6.2.6 Attributs des objets de gestion

6.2.6.1 Généralités

Les SAP de gestion de couche montrées à la Figure 25 fournissent l'accès à des informations de gestion dans les diverses couches de la suite de protocoles.

Ces informations sont représentées par des attributs définis dans les objets de gestion du DMAP, qui peuvent être contrôlés et exploités par le gestionnaire de système. Les détails des objets de gestion sont donnés en 6.2.3. Les attributs dans les objets de gestion de couche sont utilisés pour configurer les couches de protocoles et pour surveiller leur statut. Le modèle pour décrire les attributs dans tous les objets de gestion est fourni dans l'Annexe I, pour utilisation dans des extensions propriétaires et dans de futures éditions de la présente norme.

Les attributs doivent avoir un type de données qui est soit un type scalaire défini par une norme, ou une structure de données définie par une norme. Plus de détails relatifs aux attributs sont donnés en 12.6.2.

Un attribut structuré est un type spécial d'attribut qui a un type de données constitué d'une matrice de structures de données définies par une norme. Le modèle de matrice est utilisé pour permettre l'accès d'objet par le biais d'une indexation, où l'index est l'attribut clé pour l'accès à l'objet.

Les informations de gestion qu'il est nécessaire de visualiser comme étant un ensemble d'une ou plusieurs tables sont modélisées sous la forme d'attributs structurés définis dans les objets de gestion.

Les attributs définis dans des objets de gestion peuvent être accessibles en utilisant les services normalisés de lecture ou d'écriture fournis par l'ASL. De telles opérations permettent la configuration de chaque couche et la surveillance de son statut. Elles peuvent être utilisées pour récupérer, établir/modifier, et réinitialiser les valeurs des attributs. Les opérations sur des attributs sont décrites dans l'Annexe J.

6.2.6.2 Champ d'indice d'attribut structuré

Sachant que les attributs structurés sont décrits sous la forme de matrices de structures de données, il est nécessaire d'indiquer un ou plusieurs champs d'indices pour ces matrices dans la définition de chacun de ces attributs structurés. Cela se fait en incluant un * (astérisque) après le(s) nom(s) d'élément dans la table décrivant la structure des données. Le modèle pour définir une structure des données est présenté à l'Annexe I.

6.2.6.3 Métadonnées d'attribut structuré

Les attributs structurés représentent des tables d'information. Pour fournir l'accès externe au nombre d'objets contenus dans une telle table et à la capacité de cette table, des méta-attributs supplémentaires qui contiennent de telles informations sont définis pour les objets de gestion. De tels attributs représentent les métadonnées des attributs structurés correspondants.

Le type de données normalisé pour un attribut de métadonnées est présenté dans le Tableau 2.

Tableau 2 – Structure de données Metadata_attribute

Nom du type de données normalisé: Metadata_attribute		
Code du type de données normalisé: 406		
Nom de l'élément	**Identificateur de l'élément**	**Type de l'élément**
Compte (nombre de rangées à indices actuellement présentes dans l'attribut)	1	Type: Unsigned16 Classification: Static Accessibilité: Read only
Capacité (nombre de rangées que l'attribut peut contenir)	2	Type: Unsigned16 Classification: Static Accessibilité: Read only

6.2.7 Définitions des objets de gestion dans le DMAP

6.2.7.1 Device management object (objet de gestion d'appareil)

Comme montré à la Figure 24, le DMAP inclut un ensemble d'objets de gestion. L'objet de gestion d'appareil (DMO) dans le DMAP doit fournir l'accès aux attributs ayant une portée à l'échelle de l'appareil. Les attributs du DMO doivent inclure l'adresse EUI64Address principale de la DLE, un ID de fournisseur, un numéro de série, une identification de la révision courante du logiciel de communication, et la classe de source d'énergie de l'appareil. Plus de détails relatifs au DMO sont fournis en 6.2.8.1.

6.2.7.2 Alert reporting management object (objet de gestion des rapports d'alertes)

6.2.7.2.1 Généralités

L'objet de gestion de rapports d'alertes (ARMO) est utilisé pour gérer tous les rapports d'alertes de l'appareil. **Alert** (alerte) est le terme utilisé pour décrire l'action consistant à rapporter une condition d'événement ou une condition d'alarme. **Event** (événement) est le terme pour une condition transitoire (c'est-à-dire sans état), utilisé pour rapporter lorsque quelque chose s'est produit. **Alarme** est le terme utilisé pour une condition qui maintient l'état jusqu'à ce que la condition soit éliminée, qui est rapportée à la suite d'un changement d'état. Les alertes, y compris les événements et les alarmes, sont envisagées être d'une grande utilité pour gérer un réseau conforme à la présente norme.

Il doit y avoir tout au plus un seul ARMO par appareil. Les alarmes et les événements doivent être rapportés au moyen de l'ARMO. Lorsqu'une alerte est déclenchée, elle indique une situation significative qui a besoin d'être rapportée. L'ARMO doit encapsuler le rapport, gérer les temporisations et les répétitions de tentative et étrangler les rapports d'alertes issus de l'appareil.

L'ARMO fonctionne comme un proxy d'alertes pour les objets présents dans l'appareil. Toutes les alertes générées par n'importe quel objet présent dans un appareil doivent être envoyées seulement par l'ARMO qui est un objet de gestion qui est une partie intégrante du DMAP. L'APDU de données Alert doit indiquer dans son en-tête d'APDU que l'initiateur de la communication est l'objet ARMO et le DMAP de l'appareil rapportant l'alerte. L'objet et l'UAP qui sont à l'origine de l'APDU d'alerte réelle doivent être identifiés dans le contenu du rapport Alert plutôt que dans les en-têtes des APDU.

Chaque alerte doit être acquittée par l'appareil recevant l'alerte. Chaque acquittement d'alerte doit être adressée à l'ARMO de l'appareil qui était à l'origine de l'alerte. Les alertes sont promptement rapportées et précisément horodatées en utilisant des rapports d'alertes en file d'attente. Les rapports d'alertes en file d'attente impliquent que l'appareil détectant l'alerte rapporte la condition en utilisant le flux de communication S/S et reçoive en retour une DPDU ACK/NAK.

NOTE L'intention du fait de spécifier l'ARMO dans la présente norme est de séparer la détection d'alerte de la gestion des rapports relatifs à la condition d'alerte. A la différence de certains protocoles hérités orientés filaires, la présente norme consolide les alertes localement afin de réduire au maximum la messagerie externalisée et la consommation d'énergie.

Le modèle d'alerte utilisé dans la présente norme est décrit en 12.8.

Les interfaces entre l'ARMO et tous les autres objets, tant dans les UAP que dans le DMAP, sont internes à un appareil et ne sont pas spécifiées dans la présente norme

6.2.7.2.2 Types d'alertes

Les classes d'alertes, les sens d'alertes et les priorités d'alertes sont définies en 12.11. La catégorie d'alertes indique si l'alerte est une alerte de diagnostics d'appareil, une alerte de diagnostics de communication, une alerte de sécurité ou une alerte de processus. Le type d'alerte fournit des informations complémentaires relatives à l'alerte, spécifiques à la catégorie d'alertes et spécifiques à l'objet d'application générant l'alerte.

Le Tableau 3 fournit les types d'alertes pour les catégories d'alertes de la catégorie d'alertes de diagnostics de communication. Le Tableau 4 fournit les types d'alertes pour la catégorie d'alertes de sécurité. Le Tableau 5 fournit les types d'alertes pour la catégorie d'alertes de diagnostics d'appareil. Le Tableau 6 fournit les types d'alertes pour la catégorie d'alertes de processus.

Tableau 3 – Types d'alertes pour la catégorie diagnostics de communication

Type d'alerte	Catégorie d'alerte: Diagnostics de communication					
	ARMO	ASLMO	DLMO	NLMO	TLMO	DMO
0	Alarm_Recovery_Start; voir Tableau 8	Malformed APDUCommunicationAlert; voir 12.19.5	DL_Connectivity; voir 9.6.1	NL Dropped PDU; voir 10.4.3	IllegalUseOfPort; voir 11.6.2.5.4	Device_Power_Status_Check; voir 6.2.8.1.2
1	Alarm_Recovery_End; voir Tableau 8	-	Neighbor Discovery; voir 9.6.2	-	TPDUonUnregisteredPort; voir 11.6.2.5.4	Device_Restart; voir 6.2.8.1.2
2	-	-	-	-	TPDUoutOdSecurityPolicies; voir 11.6.2.5.4	-

Tableau 4 – Types d'alertes pour la catégorie d'alertes de sécurité

Type d'alerte	Catégorie d'alerte: Sécurité		
	ARMO	DSMO	DPO
0	Alarm_Recovery_Start; voir Tableau 8	Security_MPDU_Fail_Rate_Exceeded; voir 7.11.4	Not_On_Whitelist_Alert; voir Tableau 374
1	Alarm_Recovery_End; voir Tableau 8	Security_TPDU_Fail_Rate_Exceeded; voir 7.11.4	Inadequate_Join_Capability_Alert; voir Tableau 374
2	-	Security_Key_Update_Fail_Rate_Exceeded; voir 7.11.4	-

Tableau 5 – Types d'alertes pour la catégorie d'alertes de diagnostics d'appareil

Type d'alerte	Catégorie d'alerte: Diagnostics d'appareil
	ARMO
0	Alarm_Recovery_Start; voir Tableau 8
1	Alarm_Recovery_End; voir Tableau 8

Tableau 6 – Types d'alertes pour la catégorie d'alertes de processus

Type d'alerte	Catégorie d'alerte: Processus				
	ARMO	IA	AO	BI	BO
0	Alarm_Recovery_Start; voir Tableau 8	Voir 12.19.7	Voir 12.19.7	Voir 12.19.7	Voir 12.19.7
1	Alarm_Recovery_End; voir Tableau 8	-	-	-	-

6.2.7.2.3 Maître d'alertes

Les alertes doivent être envoyées à des objets récepteurs d'alertes. Les objets récepteurs d'alertes sont définis en 12.15.2.3. Chaque catégorie d'alertes peut avoir un objet récepteur d'alertes différent résidant dans un appareil différent. Les appareils qui reçoivent ces alertes sont appelés maîtres d'alertes.

L'accès de DMAP est souvent limité au SMAP présent dans le gestionnaire de système. Dans une exception à ce principe général, les maîtres d'alertes sont autorisés à accéder à l'objet ARMO présent dans le DMAP. L'accès de DMAP par les maîtres d'alertes doit être limité à l'ARMO, à moins que le maître d'alertes n'utilise la session DMAP-SMAP établie lorsque l'appareil avait rejoint le réseau. Les maîtres d'alertes auxquels l'appareil est configuré pour envoyer des alertes sont énumérés dans le Tableau 7.

6.2.7.2.4 File d'attente d'alertes

Les alertes appartenant à chaque catégorie sont supposées être placées dans une file d'attente interne fournie par catégorie dans l'appareil. Les deux types d'alertes, événements (sans état) et alarmes (à états), seront placés dans la même file d'attente, filtrée par catégorie. La file d'attente est nécessaire pour assurer une livraison garantie des alertes au maître d'alertes. Chaque alerte devant être rapportée au maître d'alertes est placée dans cette file d'attente de rapports.

Il convient que la taille de la file d'attente soit assez grande pour contenir simultanément tous les événements ainsi que toutes les conditions d'alarme possibles afin d'aider à la récupération d'alarmes sans perdre la moindre alarme.

Bien que placés dans la même file d'attente, les événements et les alarmes auront des priorités différentes. L'appareil doit rapporter un événement de plus haute priorité avant un événement de plus basse priorité. Pour les alarmes, la file d'attente est vidée séquentiellement; l'alarme la plus ancienne est rapportée en premier. Lorsque la file d'attente est pleine et une nouvelle alarme est présentée, l'alarme la plus ancienne est exclue de la file d'attente indépendamment de son état de rapport.

6.2.7.2.5 Modèles d'états d'alertes

Les tableaux d'états et transitions d'états pour les alarmes et les événements sont donnés en 12.9 et en 12.10.

6.2.7.2.6 Récupération d'alarme

Il est souvent utile d'être capable de récupérer toutes les alarmes actuellement actives au sein d'un appareil. Le besoin de récupération d'alarme se fait sentir toutes les fois qu'une connexion à l'appareil est perdue pendant une durée ou toutes les fois qu'un maître d'alertes ordonne une récupération d'alarme.

La récupération d'alarme est constituée de l'ensemble suivant d'activités:

- Le maître d'alertes ordonne une récupération d'alarme en utilisant la méthode Alarm_Recovery de l'ARMO. Cette méthode est décrite dans le Tableau 9.
- L'ARMO envoie une alerte de début de récupération au maître d'alertes, qui indique que l'ARMO a reçu une commande de récupérer des alarmes et que ces alarmes actives suivront.

NOTE Le processus pour envoyer de nouveau ces alarmes actives au sein d'un appareil n'est pas spécifié.

- L'ARMO envoie une alerte de fin de récupération au maître d'alertes.

L'ARMO est chargé de générer des alertes de début et de fin de récupération d'alarme et de coordonner le processus de récupération d'alarme avec les objets d'application résidant au sein de l'appareil.

6.2.7.2.7 Attributs, alertes et méthodes de l'objet de gestion de rapports d'alertes

Les attributs de l'ARMO sont définis dans le Tableau 7.

Tableau 7 – Attributs ARMO *(1 de 4)*

Nom du type d'objet normalisé: Alert reporting management object (ARMO, objet de gestion des rapports d'alertes)				
Identificateur du type d'objet normalisé: 126				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Alert_Master_Device_ Diagnostics	1	Maître d'alertes pour les alertes appartenant à la catégorie d'alertes de diagnostics d'appareil	Type: Point d'extrémité de communication d'alertes Classification: Static Accessibilité: Read/write Plage valide: Voir 12.16.3.5	Généralement défini sur une passerelle correspondant aux informations de l'appareil, mais peut être remplacé par tout autre appareil conforme à la norme doté d'une adresse IPv6Address valide [a]
Confirmation_Timeout _ Device_Diagnostics	2	Temporisation d'attente pour l'acquittement d'une alarme de diagnostics d'appareil qui a été envoyée au maître d'alertes	Type: Integer16 Classification: Static Accessibilité: Read/write Valeur par défaut: 10	Temporisation indépendante vis-à-vis de la proximité au maître d'alertes. Une valeur de $N > 0$ spécifie une durée de N s tandis qu'une valeur de $N < 0$ spécifie une durée de -1/N s. $N = 0$ n'est pas permise [b]
Alerts_Disable_Devic e_ Diagnostics	3	Commande permettant d'activer et de désactiver toutes les alertes de diagnostics d'appareil	Type: Boolean8 Classification: Static Accessibilité: Read/write Valeur par défaut: false	false = activer, true = désactiver

Tableau 7 *(2 de 4)*

Nom du type d'objet normalisé: Alert reporting management object (ARMO, objet de gestion des rapports d'alertes)				
Identificateur du type d'objet normalisé: 126				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Alert_Master_Comm_ Diagnostics	4	Maître d'alertes pour les alertes appartenant à la catégorie d'alertes de diagnostics de communication	Type: Point d'extrémité de communication d'alertes Classification: Static Accessibilité: Read/write Plage valide: Voir 12.16.3.5	Généralement défini sur le gestionnaire de système correspondant à l'appareil, mais peut être remplacé par tout autre appareil conforme à la norme doté d'une adresse IPv6Address valide. L'appareil doit définir cet attribut sur son gestionnaire de système après avoir rejoint le réseau; voir 6.3.7.2
Confirmation_Timeout _ Comm_Diagnostics	5	Temporisation d'attente pour l'acquittement d'une alarme de diagnostics de communication qui a été envoyée au maître d'alertes	Identique à l'attribut 2	Identique à l'attribut 2
Alerts_Disable_Comm _ Diagnostics	6	Commande permettant d'activer et de désactiver toutes les alertes de diagnostics de communication	Type: Boolean8 Classification: Static Accessibilité: Read/write Valeur par défaut: false	false = activer, true = désactiver
Alert_Master_Security	7	Maître d'alertes pour les alertes appartenant à la catégorie d'alertes de sécurité	Type: Point d'extrémité de communication d'alertes Classification: Static Accessibilité: Read/write Plage valide: Voir 12.16.3.5	Généralement défini sur le gestionnaire de système/sécurité correspondant à l'appareil, mais peut être remplacé par tout autre appareil conforme à la norme doté d'une adresse IPv6Address valide. L'appareil doit définir cet attribut sur son gestionnaire de sécurité après avoir rejoint le réseau; voir 6.3.7.2
Confirmation_Timeout _Security	8	Temporisation d'attente pour l'acquittement d'une alarme de sécurité qui a été envoyée au maître d'alertes	Identique à l'attribut 2	Identique à l'attribut 2

Tableau 7 *(3 de 4)*

Nom du type d'objet normalisé: Alert reporting management object (ARMO, objet de gestion des rapports d'alertes)				
Identificateur du type d'objet normalisé: 126				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Alerts_Disable_Security	9	Commande permettant d'activer et de désactiver toutes les alertes de sécurité	Type: Boolean8 Classification: Static Accessibilité: Read/write Valeur par défaut: false	false = activer, true = désactiver
Alert_Master_Process	10	Maître d'alertes pour les alertes appartenant à la catégorie d'alertes de processus	Type: Point d'extrémité de communication d'alertes Classification: Static Accessibilité: Read/write Plage valide: Voir 12.16.3.5	Généralement défini sur une passerelle correspondant aux informations de l'appareil, mais peut être remplacé par tout autre appareil conforme à la norme doté d'une adresse IPv6Address valide[a]
Confirmation_Timeout_Process	11	Temporisation d'attente pour l'acquittement d'une alarme de processus qui a été envoyée au maître d'alertes	Identique à l'attribut 2	Identique à l'attribut 2
Alerts_Disable_Process	12	Commande permettant d'activer et de désactiver toutes les alertes de processus	Type: Boolean8 Classification: Static Accessibilité: Read/write Valeur par défaut: false	false = activer, true = désactiver
Comm_Diagnostics_Alarm_Recovery_AlertDescriptor	13	Utilisé pour changer la priorité des événements d'alertes de début et de fin de récupération d'alarme (Tableau 8) appartenant à la catégorie d'alertes de diagnostics de communication; ces événements peuvent également être activés ou désactivés	Type: Alert report descriptor Classification: Static Accessibilité: Read/write Valeur par défaut: [false, 3] Plage valide: Voir 12.16.3.7	—

Tableau 7 *(4 de 4)*

Nom du type d'objet normalisé: Alert reporting management object (ARMO, objet de gestion des rapports d'alertes)				
Identificateur du type d'objet normalisé: 126				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Security_Alarm_Recovery_ AlertDescriptor	14	Utilisé pour changer la priorité des événements d'alertes de début et de fin de récupération d'alarme (Tableau 8) appartenant à la catégorie d'alertes de sécurité; ces événements peuvent également être activés ou désactivés	Type: Alert report descriptor Classification: Static Accessibilité: Read/write Valeur par défaut: [false, 3] Plage valide: Voir 12.16.3.7	—
Device_Diagnostics_Alarm_ Recovery_AlertDescriptor	15	Utilisé pour changer la priorité des événements d'alertes de début et de fin de récupération d'alarme (Tableau 8) appartenant à la catégorie d'alertes de diagnostics d'appareil; ces événements peuvent également être activés ou désactivés	Type: Alert report descriptor Classification: Static Accessibilité: Read/write Valeur par défaut: [false, 3] Plage valide: Voir 12.16.3.7	—
Process_Alarm_Recovery_ AlertDescriptor	16	Utilisé pour changer la priorité des événements d'alertes de début et de fin de récupération d'alarme (Tableau 8) appartenant à la catégorie d'alertes de processus; ces événements peuvent également être activés ou désactivés	Type: Alert report descriptor Classification: Static Accessibilité: Read/write Valeur par défaut: [false, 3] Plage valide: Voir 12.16.3.7	—
Réservé pour les éditions futures de la présente norme	17..63	-	-	—
[a] Ces informations sont censées être configurées par l'application hôte après que l'appareil a rejoint le réseau. [b] Toutes les alarmes exigent un acquittement.				

Les alertes de l'ARMO sont définies dans le Tableau 8.

Tableau 8 – Alertes de l'ARMO

Nom du type d'objet normalisé: Alert reporting management object (ARMO, objet de gestion des rapports d'alertes)				
Identificateur du type d'objet normalisé: 126				
Description de l'alerte: Evénements d'alertes de début et de fin de récupération d'alarmes appartenant à toutes les catégories				
Classe d'alertes (Enumerated: alarme ou événement)	**Catégorie d'alertes (Enumerated: diagnostic d'appareil, diagnostic de comm., sécurité ou processus)**	**Type d'alertes (Enumerated: en fonction de la catégorie d'alertes)**	**Priorité d'alertes (Enumerated: urgent, haut, moyen, faible, journal)**	**Description de la valeur incluse avec l'alerte**
0 = Event	1 = Comm. diagnostics	0 = Alarm_Recovery_Start	3 = Low	Générée par l'ARMO pour le maître d'alertes de diagnostics de communication indiquant que la commande de récupération d'alarme a été reçue; toutes les alarmes de diagnostics de communication en cours sont rapportées après que cet événement a été déclenché
0 = Event	1 = Comm. diagnostics	1 = Alarm_Recovery_End	3 = Low	Générée par l'ARMO pour le maître d'alertes de diagnostics de communication indiquant que le processus de récupération d'alarme est terminé
0 = Event	2 = Security	0 = Alarm_Recovery_Start	3 = Low	Générée par l'ARMO pour le maître d'alertes de sécurité indiquant que la commande de récupération d'alarme a été reçue; toutes les alarmes de sécurité en cours sont rapportées après que cet événement a été déclenché
0 = Event	2 = Security	1 = Alarm_Recovery_End	3 = Low	Générée par l'ARMO pour le maître d'alertes de sécurité indiquant que le processus de récupération d'alarme est terminé
0 = Event	0 = Device diagnostics	0 = Alarm_Recovery_Start	3 = Low	Générée par l'ARMO pour le maître d'alertes de diagnostics d'appareil indiquant que la commande de récupération d'alarme a été reçue; toutes les alarmes de diagnostics d'appareil en cours sont rapportées après que cet événement a été déclenché
0 = Event	0 = Device diagnostics	1 = Alarm_Recovery_End	3 = Low	Générée par l'ARMO pour le maître d'alertes de diagnostics d'appareil indiquant que le processus de récupération d'alarme est terminé
0 = Event	3 = Process	0 = Alarm_Recovery_Start	3 = Low	Générée par l'ARMO pour le maître d'alertes de processus indiquant que la commande de récupération d'alarme a été reçue; toutes les alarmes de processus en cours sont rapportées après que cet événement a été déclenché

Nom du type d'objet normalisé: Alert reporting management object (ARMO, objet de gestion des rapports d'alertes)				
Identificateur du type d'objet normalisé: 126				
Description de l'alerte: Evénements d'alertes de début et de fin de récupération d'alarmes appartenant à toutes les catégories				
Classe d'alertes (Enumerated: alarme ou événement)	**Catégorie d'alertes (Enumerated: diagnostic d'appareil, diagnostic de comm., sécurité ou processus)**	**Type d'alertes (Enumerated: en fonction de la catégorie d'alertes)**	**Priorité d'alertes (Enumerated: urgent, haut, moyen, faible, journal)**	**Description de la valeur incluse avec l'alerte**
0 = Event	3 = Process	1 = Alarm_Recovery_End	3 = Low	Générée par l'ARMO pour le maître d'alertes de processus indiquant que le processus de récupération d'alarme est terminé

La méthode de l'ARMO utilisée pour récupérer des alarmes des différentes catégories doit être telle que définie dans le Tableau 9.

Tableau 9 – Méthode Alarm_Recovery

Nom du type d'objet normalisé: Alert reporting management object (ARMO, objet de gestion des rapports d'alertes)				
Identificateur du type d'objet normalisé: 126				
Nom de la méthode	**ID de la méthode**	**Description de la méthode**		
Alarm_Recovery	1	Méthode pour récupérer des alarmes qui appartiennent à la catégorie mentionnée dans l'argument d'entrée		
	Arguments d'entrée			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	Alert_Category	Data type: Unsigned8	Valeurs nommées: 0: diagnostic d'appareil; 1: diagnostic de communication; 2: sécurité; 3: processus.
	Arguments de sortie			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	-	-	-	-

6.2.7.3 Upload/download object (objet téléchargement montant/téléchargement descendant)

Les attributs, les méthodes et les diagrammes d'état de l'UDO dans le DMAP doivent être selon la définition donnée en 12.15.2.4. L'identificateur d'objet de l'UDO dans le DMAP doit être 8.

Un objet de téléchargement montant/téléchargement descendant(UDO) est utilisé pour télécharger, en montant ou en descendant, de grands blocs d'informations vers/depuis un appareil. L'UDO peut être utilisé pour prendre en charge le téléchargement descendant d'une nouvelle version du firmware de communications ou des données. L'UDO maintient des informations de contrôle de révision. L'UDO est décrit en 12.15.2.4.

Le processus de mise à niveau de firmware utilisé par le gestionnaire de système pour les mises à niveau de firmware par liaison radio est décrit en 6.3.6. Les méthodes et les attributs de l'objet de téléchargement montant/téléchargement descendant dans le DMAP de l'appareil peuvent être utilisés pour envoyer des mises à jour de firmware vers l'appareil.

Le processus de mise à niveau de firmware peut inclure un mécanisme de basculement qui spécifie l'heure de basculement (après livraison de la mise à jour), point auquel les appareils

commencent à utiliser le nouveau firmware. L'attribut CutoverTime dans l'UDO doit être utilisé pour indiquer cette heure de basculement ("cut-over time"). L'heure de basculement utilise le sens partagé de temps qui a été configuré par le gestionnaire de système.

Un support est fourni pour les mises à jour spécifiques à un fournisseur, spécifiques à un modèle d'appareil et spécifiques à une instance d'appareil. Avant le basculement, l'objet de téléchargement montant/téléchargement descendant de l'appareil peut accomplir des vérifications de sécurité sur une mise à jour reçue pour s'assurer qu'une mise à jour est appropriée pour un type d'appareil spécifique. Etant donné que toute la communication a lieu entre des objets d'application, une telle mise à jour est protégée par le mécanisme de sécurité de bout en bout de la TL. En outre, la mise à jour de firmware peut utiliser des mécanismes de sécurité pour authentifier la mise à jour. Comme partie intégrante du processus de mise à niveau de firmware, l'objet de téléchargement montant/téléchargement descendant de l'appareil peut être équipé des informations appropriées relatives à l'étiquetage, au contrôle des versions et à la sécurité normalisés pour ces vérifications par l'application de mise à jour d'hôte. Ces vérifications peuvent être spécifiques à un fournisseur et ne sont pas spécifiées par la présente norme.

NOTE L'UDO dans le DMAP pour mettre à jour un firmware est décrit ici. Les détails relatifs aux objets généraux de téléchargement montant/téléchargement descendant qui sont disponibles pour une utilisation par des procédés applicatifs généraux sont donnés en 12.15.2.4.

6.2.7.4 Layer management objects (objets de gestion de couche)

L'ensemble d'objets au sein du DMAP doit inclure des objets représentant l'accès à chacun des SAP de gestion de couche. Ces objets comprennent:

- L'objet de gestion de sous-couche d'application (ASLMO), qui fournit l'accès à l'ASMSAP.
- L'objet de gestion de TL (TLMO), qui fournit l'accès au TMSAP.
- L'objet de gestion de NL (NLMO), qui fournit l'accès au NMSAP.
- L'objet de gestion de liaison de données (DLMO), qui fournit l'accès au DMSAP.

Les services, définis par les définitions de SAP de couche, sont reflétés dans les caractéristiques des objets de gestion afin qu'effectivement, les services rendus disponibles au niveau des SAP de gestion deviennent accessibles à distance en utilisant des mécanismes normalisés de communication sécurisés. Génériquement, les SAP de gestion permettent de lire et écrire des attributs, d'invoquer des méthodes, et de rapporter des événements. Les diverses spécifications de couche spécifient les caractéristiques exactes qui sont disponibles sur chacune de ces SAP. En effet, ces spécifications définissent les objets de gestion de couche. Voir 6.2.8.2 pour plus de détails relatifs aux objets de gestion de couche.

6.2.7.5 Device security management object (objet de gestion de sécurité d'appareil)

Le DMAP doit inclure un objet de gestion de sécurité d'appareil (DSMO) qui fournit l'accès judicieusement limité aux fonctions de gestion de degré de sécurité d'appareil. Le DSMO gère le support de clé de sécurité et les opérations cryptographiques. Les détails de cet objet sont fournis en 7.11.

6.2.7.6 Device provisioning object (objet de configuration d'appareil)

Le DMAP doit inclure un objet de configuration d'appareil (DPO) qui est accessible au cours du processus de configuration de l'appareil. Plus de détails relatifs aux attributs, aux méthodes et aux alertes du DPO sont fournis en 13.9.

6.2.7.7 Health reports concentrator object (objet concentrateur de bulletins de santé)

Le DMAP doit inclure un objet concentrateur de rapports de santé (HRCO) qui peut être configuré par le gestionnaire de système pour permettre la publication périodique de rapports

de santé d'appareil. Les rapports de santé d'appareil peuvent consister en une publication périodique d'un ou plusieurs attributs issus d'objets de gestion dans le DMAP.

Les attributs du HRCO sont conformes à la définition de l'objet concentrateur donnée en 12.15.2.5. L'identificateur d'objet du HRCO dans le DMAP doit être 10. Les attributs CommunicationEndPoint et Array of ObjectAttributeIndexAndSize du HRCO sont utilisés par le gestionnaire de système pour établir des publications périodiques de rapports de santé à partir de l'appareil. Le gestionnaire de système peut choisir d'inclure n'importe quel attribut de n'importe quel objet de gestion dans de tels rapports de santé qui sont utilisés pour la surveillance des performances du système. La surveillance des performances du système est décrite en 6.3.7.

6.2.8 Fonctions de gestion d'appareil et de gestion de couche

6.2.8.1 Fonctions de gestion d'appareil

6.2.8.1.1 Généralités

Des capacités de gestion d'appareil sont fournies principalement par l'intermédiaire de l'accès aux attributs de DMO et à l'invocation des méthodes. Le DMO contient les attributs critiques de portée à l'échelle de l'appareil qui doivent être disponibles dans des tous les appareils. Les réalisateurs de produit peuvent étendre la liste d'attributs au-delà des attributs exigés qui sont décrits ci-dessous.

Certains attributs disponibles par l'intermédiaire du DMO peuvent également être disponibles comme attribut d'un objet de gestion de couche particulier. Dans ce cas, une modification de la valeur d'un tel attribut doit être reflété dans l'attribut correspondant.

Le DMO doit fournir des informations de temps système à d'autres objets de gestion; le DMO peut obtenir ces informations de temps système en interagissant avec la DL de l'appareil et/ou du gestionnaire de système ou à partir d'une autre source, telle qu'un récepteur GPS au sein de l'appareil. Le chronométrage par la DL est décrit en 9.1.9. Le rôle du gestionnaire de système dans le maintien du temps à travers le réseau est décrit en 6.3.10.

L'établissement, la modification et la résiliation de contrats pour un appareil doivent être gérés par son DMO. Les contrats sont décrits en 6.3.11.2.

Les attributs du DMO sont définis dans le Tableau 10.

Tableau 10 – Attributs du DMO *(1 de 9)*

Nom du type d'objet normalisé: Device management object (DMO, objet de gestion d'appareil)				
Identificateur du type d'objet normalisé: 127				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
EUI64	1	Identificateur de 64 bits de l'appareil	Type: EUI64Address Classification: Constant Accessibilité: Read only Valeur par défaut: 0x0000 0000 0000 0001	Il doit s'agir d'une adresse EUI64Address globale unique. Cet attribut est un doublon des attributs correspondants dans le DLMO et le NLMO.
DL16Address	2	Identificateur de 16 bits de l'appareil, unique dans son sous-réseau D	Type: DL16Address Classification: Static Accessibilité: Read/write Valeur par défaut: 0 Plage valide: 0: adresse non affectée; 1..0x7FFF: adresse de monodiffusion	Adresse unique dans le sous-réseau D de l'appareil; assignée par le gestionnaire de système. Cet attribut est un doublon de l'attribut correspondant dans le DLMO et le NLMO. Cet attribut est configuré par le gestionnaire de système au cours du processus de rattachement de l'appareil au sous-réseau.
IPv6Address	3	Adresse IPv6Address assignée par le gestionnaire de système	Type: IPv6Address Classification: Static Accessibilité: Read/write Valeur par défaut: 0 Plage valide: 0: adresse non affectée; autre valeur avec réglage sur bit de poids fort: adresse de monodiffusion.	Adresse de réseau de l'appareil unique dans le réseau et utilisée par l'application pour identifier des appareils à travers le réseau. Cet attribut est un doublon des attributs correspondants dans le DLMO et le NLMO. Cet attribut est configuré par le gestionnaire de système au cours du processus de rattachement de l'appareil au sous-réseau.

Tableau 10 *(2 de 9)*

Nom du type d'objet normalisé: Device management object (DMO, objet de gestion d'appareil)				
Identificateur du type d'objet normalisé: 127				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Device_Role_Capability	4	Rôle(s) que l'appareil est capable de jouer dans le réseau; les rôles sont définis en 5.2.6.2	Type: BitArray16 Classification: Constant Accessibilité: Read only	Cet attribut doit être envoyé au gestionnaire de système au cours du processus de rattachement de l'appareil au sous-réseau; voir 6.3.9.2. Voir 5.2.6.2 pour les rôles acceptables et leurs descriptions. Indices nommés: 0: E/S; 1: routeur; 2: routeur dorsal; 3: passerelle; 4: gestionnaire de système; 5: gestionnaire de sécurité; 6: source de temps système; 7: appareil de configuration; 8..15: réservés (doivent être mis à 0)
Assigned_Device_Role	5	Rôle(s) de l'appareil tel(s) qu'assigné(s) par le gestionnaire de système; les rôles sont définis en 5.2.6.2	Type: BitArray16 Classification: Static Accessibilité: Read/write	Cet attribut doit être écrit par le gestionnaire de système au cours du processus de rattachement de l'appareil au sous-réseau; voir 6.3.9.2. Voir 5.2.6.2 pour les rôles acceptables et leurs descriptions. Le rôle assigné à un appareil ne doit pas dépasser ses capacités, telles que spécifiées dans l'attribut 4. Les indices de la matrice de bits sont identiques à ceux de l'attribut 4.
Vendor_ID	6	Identification du fournisseur de l'appareil lisible par l'homme	Type: VisibleString16 Classification: Constant Accessibilité: Read only	Assigné par le fournisseur au cours de la fabrication de l'appareil
Model_ID	7	Identification du modèle de l'appareil lisible par l'homme	Type: VisibleString16 Classification: Constant Accessibilité: Read only	Assigné par le fournisseur au cours de la fabrication de l'appareil
Tag_Name	8	Nom TAG de l'appareil	Type: VisibleString16 Classification: Static Accessibilité: Read/write	Attribué par l'utilisateur

Tableau 10 *(3 de 9)*

Nom du type d'objet normalisé: Device management object (DMO, objet de gestion d'appareil)				
Identificateur du type d'objet normalisé: 127				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Serial_Number	9	Numéro de série de l'appareil	Type: VisibleString16 Classification: Constant Accessibilité: Read only	Assigné par le fournisseur au cours de la fabrication de l'appareil
Power_Supply_Status	10	Informations de statut de l'alimentation en énergie de l'appareil	Type: Unsigned8 Classification: Dynamic Accessibilité: Read/write	Valeurs nommées: 0: alimenté par secteur; 1: alimenté par pile, plus de 75 % de capacité restante; 2: alimenté par pile, entre 25 % et 75 % de capacité restante; 3: alimenté par pile, moins de 25 % de capacité restante
Device_Power_Status_Check_AlertDescriptor	11	Utilisé pour changer la priorité d'alerte Device_Power_Status_Check (décrit dans le Tableau 11); cette alerte peut également être activée ou désactivée	Type: Alert report descriptor Classification: Static Accessibilité: Read/write Valeur par défaut: [false, 8]	—
DMAP_State	12	Statut de DMAP	Type: Unsigned8 Classification: Dynamic Accessibilité: Read only Valeur par défaut: 1: actif	Le diagramme d'états de DMAP est le même que le diagramme d'états d'UAP donné en 12.15.2.2.3. Valeurs nommées: 0: inactif; 1: actif; 2: en panne

Tableau 10 *(4 de 9)*

Nom du type d'objet normalisé: Device management object (DMO, objet de gestion d'appareil)				
Identificateur du type d'objet normalisé: 127				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Join_Command	13	Commande informant l'appareil de rejoindre le système, de se redémarrer lui-même et de se rattacher de nouveau, ou se réinitialiser aux valeurs d'usine par défaut	Type: Unsigned8 Classification: Static Accessibilité: Read/write Valeur par défaut: 0: none	L'utilisation de cet attribut est décrite en 6.3.9. La valeur 0 ne doit pas être indiquée dans une demande d'écriture. Seul l'appareil de configuration est censé être capable d'émettre la commande Join_Command = 1 (join and start), car l'appareil n'a pas encore rejoint le réseau et n'est donc accessible par aucun autre appareil. WarmRestart doit préserver les données des attributs statiques et constants, y compris les contrats et les clés T. RestartAsProvisioned correspond à l'état configuré de l'appareil dans lequel l'appareil retient seulement les informations qui sont reçues pendant son étape de configuration. La réinitialisation aux valeurs d'usine par défaut correspond à la phase d'appareil configuré montrée à la Figure 5. Valeurs nommées: 0: aucun choix; 1: “join and start”; 2: WarmRestart; 3: RestartAsProvisioned; 4: réinitialisation aux valeurs d’usine par défaut

Tableau 10 *(5 de 9)*

Nom du type d'objet normalisé: Device management object (DMO, objet de gestion d'appareil)				
Identificateur du type d'objet normalisé: 127				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Static_Revision_Level	14	Niveau de révision des données statiques associées à tous les objets de gestion	Type: Unsigned32 Classification: Dynamic Accessibilité: Read only Valeur par défaut: 0: none	Le niveau de révision est incrémenté chaque fois qu'une valeur d'attribut statique dans n'importe quel objet de gestion change; la valeur repasse par zéro lorsque la limite est atteinte; la valeur se réinitialise chaque fois que l'appareil est réinitialisé aux valeurs d'usine par défaut, soit Join_Command = 4 (reset to factory defaults)
Restart_Count	15	Nombre de fois que l'appareil est redémarré	Type: Unsigned16 Classification: Static Accessibilité: Read only Valeur par défaut: 0	Le redémarrage de l'appareil peut être dû à un remplacement de batterie, une commande de redémarrage à chaud, un téléchargement descendant de firmware, une défaillance de liaison; la valeur repasse par zéro si la valeur maximale est atteinte; la valeur se réinitialise à 0 lorsque l'appareil est réinitialisé aux valeurs d'usine par défaut
Uptime	16	Compteur de faible exactitude pour compter les secondes écoulées depuis le dernier redémarrage de l'appareil	Type: Unsigned32 Classification: Dynamic Accessibilité: Read only Valeur par défaut: 0	Unités en secondes; la valeur se réinitialise à 0 si l'appareil redémarre
Device_Memory_Total	17	Mémoire totale de l'appareil exprimée en octets	Type: Unsigned32 Classification: Constant Accessibilité: Read only	Unités en octets
Device_Memory_Used	18	Mémoire actuellement utilisée dans l'appareil exprimée en octets	Type: Unsigned32 Classification: Dynamic Accessibilité: Read only	Unités en octets
TAI_Time	19	Temps TAI actuel	Type: TAINetworkTime Classification: Dynamic Accessibilité: Read only	La valeur est obtenue soit à partir de la DL (si l'appareil n'est pas source de temps système), soit la dorsale/source externe (si l'appareil est source de temps système ou est sur la dorsale et n'a pas une DL)

Tableau 10 *(6 de 9)*

Nom du type d'objet normalisé: Device management object (DMO, objet de gestion d'appareil)				
Identificateur du type d'objet normalisé: 127				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Comm_SW_Major_Version	20	Version majeure du logiciel de communications actuellement utilisée dans l'appareil	Type: Unsigned8 Classification: Constant Accessibilité: Read only Valeur par défaut: 0	Le numéro de version majeure de 8 bits du logiciel de communications, assigné par la présente norme, est égal à 0
Comm_SW_Minor_Version	21	Version mineure du logiciel de communications actuellement utilisée dans l'appareil	Type: Unsigned8 Classification: Constant Accessibilité: Read only	Le numéro de version mineure de 8 bits du logiciel de communications, assigné par la présente norme, est égal à 1
Software_Revision_ Information	22	Informations de révision relatives au logiciel de communications pour les numéros particuliers des versions majeure et mineure	Type: VisibleString16 Classification: Constant Accessibilité: Read only	Informations de révision assignées par le fournisseur
System_Manager_ IPv6Address	23	Adresse réseau de gestionnaire de système	Type: IPv6Address Classification: Static Accessibilité: Read/write Valeur par défaut: 0	Ces informations doivent être fournies à l'appareil soit au cours du processus de configuration, soit au cours du processus de rattachement de l'appareil au sous-réseau.
System_Manager_ EUI64Address	24	Adresse EUI64Address du gestionnaire de système	Type: EUI64Address Classification: Static Accessibilité: Read/write Valeur par défaut: 0	Ces informations doivent être fournies à l'appareil soit au cours du processus de configuration, soit au cours du processus de rattachement de l'appareil.
System_Manager_ DL16Address	25	Adresse DL16Address du gestionnaire de système dans le sous-réseau D de l'appareil	Type: DL16Address Classification: Static Accessibilité: Read/write Valeur par défaut: 0	Cet attribut doit être configuré par le gestionnaire de système au cours du processus de rattachement de l'appareil.

Tableau 10 *(7 de 9)*

Nom du type d'objet normalisé: Device management object (DMO, objet de gestion d'appareil)				
Identificateur du type d'objet normalisé: 127				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Contracts_Table	26	Tableau qui inclut les informations relatives à tous les contrats existants de l'appareil	Type: Array of Contract_Data Classification: Static Accessibilité: Read/write	Cet attribut est mis à jour lorsqu'un contrat correspondant est établi, modifié, renouvelé ou résilié; voir 6.3.11.2 pour plus de détails relatifs aux contrats et au type de données Contract_Data. Une nouvelle entrée dans la Contracts_Table doit être créée chaque fois qu'une réponse de contrat associée à la création réussie d'un nouveau contrat est reçue en provenance du gestionnaire de système. Pour plus de détails, voir 6.3.11
Contract_Request_ Timeout	27	Temporisation pour DMO avant la demande de contrat ne soit répétée	Type: Unsigned16 Classification: Static Accessibilité: Read/write	Le gestionnaire du système établit cette valeur de temporisation après que l'appareil a rejoint le réseau. Unité: s
Max_ClientServer_Retries	28	Nombre maximal de répétitions de tentative de demande client que le DMAP doit envoyer afin d'avoir une communication C/S réussie	Type: Unsigned8 Classification: Static Accessibilité: Read/write Valeur par défaut: 3 Plage valide: 0..8	Le nombre de répétitions de tentative envoyées pour un message particulier peut varier par message sur la base de la détermination du processus applicatif de l'importance du message. Par exemple, certains messages ne peuvent pas du tout faire l'objet de répétitions de tentative, et d'autres peuvent être répétés le nombre maximal de fois.
Max_Retry_Timeout_ Interval	29	Intervalle de temporisation maximal pour une demande client avant qu'elle ne soit envoyée à nouveau	Type: Unsigned16 Classification: Static Accessibilité: Read/write Valeur par défaut: 30 s	Le gestionnaire du système établit cette valeur de temporisation après que l'appareil a rejoint le réseau. Unité: s

Tableau 10 *(8 de 9)*

Nom du type d'objet normalisé: Device management object (DMO, objet de gestion d'appareil)				
Identificateur du type d'objet normalisé: 127				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
DMAP_Objects_Count	30	Nombre d'objets de gestion dans le DMAP, y compris ce DMO	Type: Unsigned8 Classification: Static Accessibilité: Read only Valeur par défaut: 1 Plage valide: > 0	Compte total d'objets de gestion tels que DLMO, NLMO, etc. dans le DMAP de cet appareil; tous les processus applicatifs dans l'appareil doivent inclure un attribut avec de telles informations
DMAP_Objects_List	31	Liste de tous les objets de gestion dans le DMAP	Type: Array of ObjectIDandType Classification: Static Accessibilité: Read Only	Liste permettant d'identifier tous les objets de gestion qui sont disponibles dans le DMAP; tous les processus applicatifs dans l'appareil doivent inclure un attribut avec de telles informations. Voir 12.16.3.10 pour les détails relatifs à ce type de données
Metadata_Contracts_Table	32	Métadonnées (compte et capacité) de l'attribut Contracts_Table	Type: Metadata_attribute	Métadonnées contenant un compte du nombre d'entrées dans le tableau et la capacité (nombre total de rangées autorisées) pour le tableau; voir 6.2.6.3 pour les détails relatifs à ce type de données
Non_Volatile_Memory_ Capability	33	Indique si l'appareil est capable de maintenir toutes les informations de DMAP qui s'inscrivent dans la classification Static en mémoire non volatile sur un cycle de mise sous/hors tension	Type: Boolean8 Classification: Constant Accessibilité: Read only	Voir 6.3.9.4.2 pour plus d'informations
Warm_Restart_Attempts_ Timeout	34	Temporisation après laquelle un appareil qui essaie de rejoindre à nouveau le réseau par un warmRestart passe à une commande restartAsProvisioned	Type: Unsigned16 Classification: Static Accessibilité: Read/write Valeur par défaut: 60	Unités en minutes; voir 6.3.9.4.2 pour plus d'informations

Tableau 10 *(9 de 9)*

Nom du type d'objet normalisé: Device management object (DMO, objet de gestion d'appareil)				
Identificateur du type d'objet normalisé: 127				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Device_Restart_AlertDescriptor	35	Utilisé pour changer la priorité d'alerte Device_Restart (décrit dans le Tableau 11); cette alerte peut également être activée ou désactivée	Type: Alert report descriptor Classification: Static Accessibilité: Read/write Valeur par défaut: [false, 8]	-
Proxy_Join_Request_Rate	36	Utilisé pour commander la fréquence maximale à laquelle un routeur proxy prendra en charge des demandes de rattachement	Type: Integer8 Classification: Static Accessibilité: Read/write Valeur par défaut: 6 Plage valide: -4..127	Intervalle exigé minimal entre les demandes de rattachement que le routeur proxy est autorisé à accepter. Une valeur de $N > 0$ spécifie une période de N s tandis qu'une valeur de $N < 0$ spécifie une période de $-1/N$ s. $N = 0$ désactive l'attribut. Ce paramètre est utilisé pour réduire l'impact des attaques de déni de service (DoS).
Réservé pour les éditions futures de la présente norme	37..63	—	—	—

6.2.8.1.2 Alertes de l'objet de gestion d'appareil

Le DMO de l'appareil doit envoyer une alerte pour indiquer un changement de son statut de puissance. Le DMO de l'appareil doit envoyer une alerte chaque fois qu'il passe par un redémarrage d'appareil. Le redémarrage d'appareil est décrit en 6.3.9.4.2.

Les alertes du DMO sont définies dans le Tableau 11.

Tableau 11 – Alertes de DMO

Nom du type d'objet normalisé: Device management object (DMO, objet de gestion d'appareil)					
Identificateur du type d'objet normalisé: 127					
Description de l'alerte: Alertes de diagnostic de communication pour indiquer que le statut d'alimentation en énergie de l'appareil a changé et pour indiquer que l'appareil a redémarré					
Classe d'alertes (Enumerated: alarme ou événement)	**Catégorie d'alertes (Enumerated: diagnostic d'appareil, diagnostic de comm., sécurité ou processus)**	**Type d'alertes (Enumerated: en fonction de la catégorie d'alertes)**	**Priorité d'alertes (Enumerated: urgent, haut, moyen, faible, journal)**	**Type de données de la valeur**	**Description de la valeur incluse avec l'alerte**
0 = Event	1 = Comm. diagnostics	0 = Device_Power_ Status_Check	8 = Medium	Type: Unsigned8	La valeur courante de l'attribut Power_Supply_ Status dans le Tableau 10 est incluse dans cette alerte
0 = Event	1 = Comm. diagnostics	1 = Device_Restart	8 = Medium	Type: N/A	Seul un appareil qui a l'attribut de DMO Non_Volatile_M emory_Capabilit y = 1 peut envoyer cette alerte pour indiquer qu'il est passé par un redémarrage à chaud

6.2.8.1.3 Méthodes de l'objet de gestion d'appareil

Les méthodes du DMO sont décrites en 6.3.9.2.2, en 6.3.11.2.10.5 et en 6.3.11.2.11.3.

6.2.8.2 Gestion de couche

6.2.8.2.1 Généralités

Chaque couche de communication au sein de la suite de protocoles a une fonctionnalité autonome de gestion de couche. Chaque fonction de gestion de couche fournit un SAP de gestion. La gestion d'appareil des couches est accomplie par l'intermédiaire de l'accès aux SAP de gestion sur chacune des couches, conformément à la Figure 25. La gestion des couches au sein d'un appareil peut être faite localement par la fonctionnalité qui réside au sein du DMO, car le DMO a accès aux SAP de gestion. En outre (voir le débat en 6.2.4), la gestion de couche peut être accomplie à distance par un gestionnaire de système.

La définition formelle de chacun des objets de gestion de couche est incluse ci-dessous. La définition des objets de gestion de couche au sein du DMAP correspond directement à la définition du SAP de gestion de couche. Cependant, il peut y avoir un besoin de limiter

l'accès distant à des caractéristiques spécifiques d'un SAP de gestion de couche donné. Ainsi, il convient que certains attributs ou certaines méthodes, bien qu'accessibles au DMO local, ne soient pas accessibles à distance. De telles restrictions, toutes les fois que nécessaires, sont spécifiées dans les spécifications de la couche. Les opérations décrites dans l'Annexe J peuvent être utilisées pour accéder à des attributs dans ces objets de gestion de couche.

6.2.8.2.2 Objet de gestion de DL

6.2.8.2.2.1 Généralités

Dans l'architecture définie par la présente norme, tous les services de gestion de couche DL, MAC et PhL sont fournis par l'intermédiaire d'un point d'accès au service de gestion de liaison de données (DMSAP) unifié. Il n'y a pas de SAP de gestion directement accessible pour la couche physique et la couche MAC, car ces couches exigent parfois que les actions de gestion soient synchrones avec le flot de données. Ainsi, le DLMO fournit les attributs des couches qui sont accessibles à distance.

6.2.8.2.2.2 Gestion de la couche physique

Les entités de gestion de couche physique sont manipulées indirectement par l'intermédiaire du DMSAP. Les attributs de DL sont relatifs à un sous-ensemble de celles qui sont définies dans l'IEEE 802.15.4, telles que décrites en 9.1.5.

6.2.8.2.2.3 Gestion de la sous-couche de commande d'accès au support

Les entités de gestion de la sous-couche MAC sont manipulées indirectement par l'intermédiaire du DMSAP. Les attributs de DL sont relatifs à un sous-ensemble de celles qui sont définies dans l'IEEE 802.15.4, telles que décrites en 9.1.5.

6.2.8.2.2.4 Gestion de DL

Les attributs et méthodes de gestion de la DL sont disponibles par l'intermédiaire du DMSAP. Les attributs, les méthodes et les alertes du DLMO sont définis en 9.4 et en 9.6.

6.2.8.2.2.5 Gestion de NL

Les attributs et méthodes de gestion de la NL sont disponibles par l'intermédiaire du NMSAP. Les attributs, les méthodes et les alertes du NLMO sont définis en 10.4.

6.2.8.2.2.6 Gestion de TL

Les attributs et méthodes de gestion de la TL sont disponibles par l'intermédiaire du TMSAP. Les attributs, les méthodes et les alertes du TLMO sont définis en 11.6.

6.2.8.2.2.7 Gestion de sécurité

Les attributs et les méthodes de gestion de sécurité d'appareil sont disponibles par l'intermédiaire du DMSAP et du TMSAP. Les attributs, les méthodes et les alertes du DSMO sont définis en 7.11.

6.2.8.2.2.8 Gestion de la sous-couche d'application

Les attributs et méthodes de gestion de la sous-couche d'application sont disponibles par l'intermédiaire de l'ASMSAP. Les attributs, les méthodes et les alertes de L'ASLMO sont définis en 12.19.

6.3 Gestionnaire de système

6.3.1 Généralités

Les fonctions du gestionnaire de système incluent la gestion de sécurité, l'allocation d'adresse, la mise à jour de logiciel, la surveillance des performances du système, la gestion d'appareil, les services de temps système et la configuration de communications incluant les services de contrat, et la gestion de redondance.

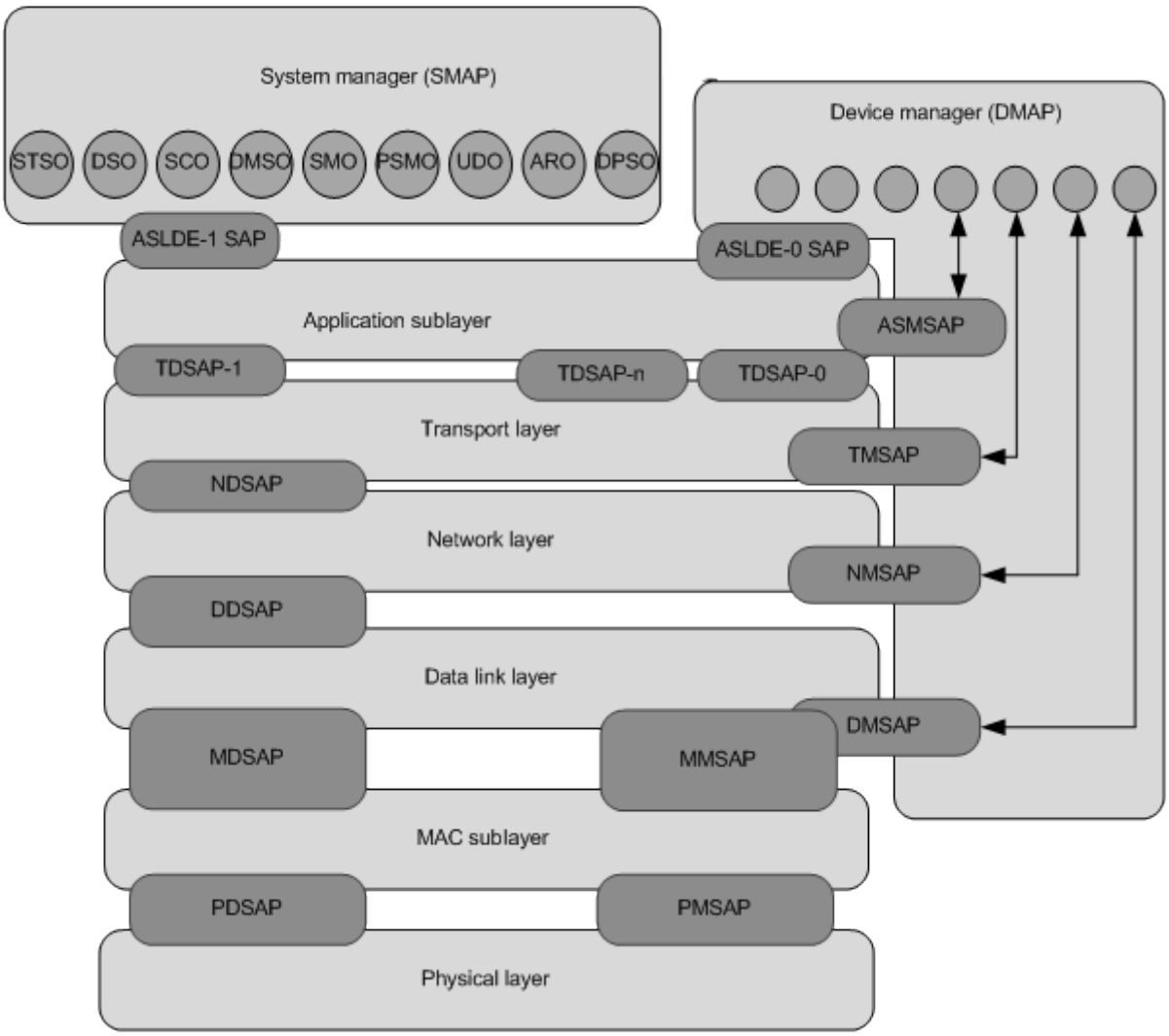
Le gestionnaire de système doit utiliser les services de l'ASL pour accéder à distance à des objets de gestion dans les DMAP des appareils conformes à la présente norme.

Le gestionnaire de système est un rôle et n'est pas attaché à une adresse physique fixe spécifique.

6.3.2 Architecture de la gestion de système

Du point de vue conceptuel, le gestionnaire de système peut être vu comme un processus applicatif tournant sur n'importe quel appareil du réseau. Un tel appareil doit être capable de prendre en charge le rôle de gestionnaire de système. Le SMAP est accessible uniquement sur un tel appareil. Le SMAP doit utiliser le SAP de la sous-couche d'application, à savoir le SAP ASLDE-1, pour communiquer avec les appareils. Ce SAP de sous-couche d'application doit correspondre au SAP de TL TDSAP-1 qui doit correspondre au numéro de port 0xF0B1.

La Figure 26 montre le gestionnaire de système qui réside dans un appareil de terrain conforme à la présente norme.



Légende

Anglais	Français
System manager (SMAP)	Gestionnaire de système (SMAP)
Device manager (DMAP)	Gestionnaire d'appareil (DMAP)
Application sub-layer	Sous-couche d'application
Transport layer	Couche transport
Network layer	Couche réseau
Data link layer	Couche liaison de données
MAC sub-layer	Sous-couche MAC
Physical layer	Couche physique

Figure 26 – Concept de l'architecture de gestionnaire de système

Comme montré à la Figure 26, TDSAP-1 doit être utilisé pour accéder aux objets de gestion dans le SMAP. La définition de ces objets de gestion de système est nécessaire pour fournir l'accès distant à ces fonctions pour les appareils dans le réseau qui sont conformes à la présente norme.

6.3.3 Types normalisés des objets de gestion de système

Le Tableau 12 inclut une liste des types d'objet de gestion de système qui sont spécifiés dans la présente norme.

Tableau 12 – Types des objets de gestion de système

Nom du type d'objet normalisé	Identificateur du type d'objet normalisé	Identificateur de l'objet normalisé	Description de l'objet
System time service object (STSO)	100	1	Cet objet facilite la gestion des informations de temps dans tout le système; voir 6.3.10
Directory service object (DSO)	101	2	Cet objet facilite la gestion des adresses de tous les appareils existants dans le réseau; voir 6.3.5
Objet de configuration de communications système (SCO)	102	3	Cet objet facilite la configuration des communications du système, y compris l'établissement, la modification et la fin de contrat; voir 6.3.11
Device management service object (DMSO)	103	4	Cet objet facilite l'attachement d'appareil, la fermeture d'appareil et la configuration des communications d'appareil; voir 6.3.9
System monitoring object (SMO)	104	5	Cet objet facilite la surveillance des performances du système; voir 6.3.7
Proxy security management object (PSMO)	105	6	Cet objet agit en tant que proxy pour le gestionnaire de sécurité; voir 6.3.4
Upload/download object (UDO)	3	7	Cet objet facilite le téléchargement de firmware/de données vers les appareils et le chargement de données à partir des appareils; voir 6.3.6
Alert-receiving object (ARO)	2	8	Cet objet reçoit toutes les alertes destinées au gestionnaire de système; voir 6.3.7
Device provisioning object (DPO)	106	9	Cet objet facilite la configuration d'appareil; voir 6.3.8
Réservé pour les éditions futures de la présente norme	107..113	—	—

Les appareils qui exigent des services de gestion de système communiquent avec les objets appropriés donnés ci-dessus.

6.3.4 Gestion de sécurité

Le gestionnaire de système s'interface avec le gestionnaire de sécurité pour générer des clés et authentifier des appareils. Le gestionnaire de sécurité est fonctionnellement séparé du gestionnaire de système afin que les politiques de sécurité puissent être communes à travers les réseaux de l'administrateur et d'autres types de réseaux. Le fait de placer le gestionnaire de sécurité fonctionnellement derrière le gestionnaire de système cache également aux appareils les divers protocoles, tels que le Kerberos, qui peuvent être utilisés par une fonction de gestion de sécurité. Plus de détails sont fournis à l'Article 7.

L'interface entre le gestionnaire de système et le gestionnaire de sécurité n'est pas spécifiée dans la présente norme. Du point de vue conceptuel, le gestionnaire de système peut être vu comme incluant un objet proxy de gestion de sécurité (PSMO). Ce PSMO transmet tous les messages relatifs à la sécurité entre le gestionnaire de sécurité et les appareils dans le réseau qui sont conformes à la présente norme. Ce PSMO peut être utilisé par le gestionnaire de sécurité pour accéder à des informations issues d'autres objets de gestion de système, telles que le temps TAI courant, s'il y a lieu. Le gestionnaire de sécurité n'a pas une adresse valide telle que définie par la présente norme; donc, les appareils qui souhaitent communiquer avec le gestionnaire de sécurité ne peuvent le faire qu'en communiquant avec le PSMO.

Les attributs, les méthodes et les alertes du PSMO sont définis à l'Article 7.

6.3.5 Adresses et allocation d'adresse

6.3.5.1 Généralités

Le gestionnaire de système a la responsabilité d'assigner des adresses aux appareils lorsqu'ils rejoignent le réseau.

6.3.5.2 Types d'adresse

Chaque appareil conforme à la présente norme doit avoir un identificateur et deux adresses comme suit:

- Chaque appareil conforme à la présente norme doit avoir un identificateur EUI64Address qui est présumé être unique au niveau global (les fournisseurs sont censés assurer l'unicité globale de ces identificateurs). Des mécanismes de reprise sur défaillance pour les rôles de passerelle, de gestionnaire de système et de gestionnaire de sécurité sont habituellement fournis par le biais de la redondance. La redondance pour les besoins de la reprise sur défaillance peut impliquer la duplication de l'identificateur EUI64Address pour les entités redondantes. Une telle duplication d'identificateur EUI64Address ne relève pas du domaine d'application de la présente norme.
- Chaque appareil conforme à la présente norme doit avoir une adresse IPv6Address par le gestionnaire de système lorsqu'il rejoint le réseau; cette adresse IPv6Address doit être unique à travers le réseau.
- Chaque appareil conforme à la présente norme qui est accessible par un sous-réseau doit avoir une adresse DL16Address dans le sous-réseau D pour son adresse IPv6Address. Cette adresse DL16Address doit être assignée par le gestionnaire de système. La portée de toute adresse DL16Address est limitée à un sous-réseau D particulier.

Plages et usages des adresses D de 16 bits:

– 0x0000: réservée par la présente norme pour indiquer qu'une adresse DL16Address n'a pas été affectée à l'appareil;

– 0x0001..0x7FFF: réservées par l'IETF RFC 4944 pour les adresses de monodiffusion 6LoWPAN;

– 0x8000..0xBFFF: réservées par l'IETF RFC 4944 pour les adresses de multidiffusion 6LoWPAN;

– 0xC000..0xCFFF: réservées par la présente norme pour les identificateurs de graphe;

– 0xD000..0xFFFD: réservées;

– 0xFFFE: réservée par l'IEEE 802.15.4e pour le coordinateur PAN local; et

– 0xFFFF: réservée par l'IEEE 802.15.4 pour la diffusion locale.

Dans la présente norme, l'adressage DL16Addressing est toujours utilisé au sein d'un sous-réseau D, excepté que les identificateurs EUI64Address sont utilisés d'une manière limitée au cours du processus de rattachement tant que l'appareil se rattachant n'a pas reçu une adresse DL16Address locale au sous-réseau D de la part du gestionnaire de système. Le processus de rattachement est décrit en 7.4.

6.3.5.3 Allocation d'adresse

Le gestionnaire de système doit allouer l'adresse IPv6Address, ainsi que l'adresse DL16Address dans le sous-réseau D, à un appareil lorsqu'il rejoint le réseau. Cela est décrit en 7.4.

Lorsqu'un appareil source qui appartient à un sous-réseau D particulier communique sur la dorsale avec un appareil de destination qui appartient à un sous-réseau D différent, le gestionnaire de système doit assigner des adresses DL16Addresses locales uniques à un sous-réseau D aux deux appareils dans les sous-réseaux D de l'un et de l'autre. De telles

adresses DL16Addresses locales pour des appareils distants (à savoir des appareils résidant dans un autre sous-réseau D) peuvent être établies par le gestionnaire de système sur une demande de contrat. (Les contrats sont décrits en 6.3.11.2.)

Lorsque la source envoie un message vers la destination, l'adresse DL16Address de la destination doit être utilisée à la NL et à la DL pour construire la NPDU et la DPDU. Ces couches peuvent utiliser le service de consultation d'annuaire fourni par le gestionnaire de système pour obtenir l'adresse DL16Address de la destination, si elle n'est pas déjà connue. Ce service est décrit en 6.3.5.4.

Les routeurs dorsaux doivent effectuer les conversions d'adresses entre l'adresse DL16Address (par sous-réseau D) et l'adresse IPv6Address pour un appareil donné. Noter que l'adresse DL16Address est utilisée seulement au sein de la DL. Une fois qu'un message atteint la dorsale, l'adresse IPv6Address complète doit être utilisée. Des exemples informatifs pour de tels scénarios sont donnés en 10.2.7.

La présente norme ne spécifie aucun mécanisme pour la façon dont le gestionnaire de système alloue les adresses IPv6Address et les adresses DL16Address. Le paragraphe 10.2.7 contient des exemples pour le routage basé sur Ethernet et le routage basé sur le bus de terrain. L'exemple du routage basé sur Ethernet décrit l'utilisation d'adresses IPv6Address basées sur IPv6. L'exemple du routage basé sur le bus de terrain décrit l'utilisation d'adresses IPv6Address qui ne sont pas basées sur IPv6.

Les appareils doivent seulement avoir une adresse IPv6Address valide. Les appareils à rattachement multiple qui exigent plusieurs adresses IPv6Address ne sont pas couverts dans la présente norme.

Les entités adressées sur la dorsale, telles que les gestionnaires de système et les passerelles, doivent également avoir des adresses DL16Address qui leur sont assignées en vue d'une utilisation dans un sous-réseau D Par conséquent, la plupart des adresses utilisées dans un sous-réseau D seront des adresses DL16Address.

6.3.5.4 Service d'annuaire

L'objet de service d'annuaire (DSO) dans le gestionnaire de système fournit les attributs nécessaires pour rechercher la conversion d'adresses entre l'adresse EUI64Address, l'adresse IPv6Address et la ou les adresses DL16Address d'un appareil donné. Des Identificateurs de sous-réseau D sont également maintenus par le DSO, mais l'allocation de ces ID de sous-réseau D n'est pas spécifiée dans la présente norme.

Les attributs du DSO sont définis dans le Tableau 13.

Tableau 13 – Attributs du DSO

Nom du type d'objet normalisé: Directory service object (DSO, objet de service d'annuaire)				
Identificateur du type d'objet normalisé: 101				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Address_Translation_Table	1	Table de conversion d'adresses contenant l'adresse EUI64Address, l'adresse IPv6Address, le ou les ID de sous-réseau D ainsi que la ou les adresses DL16Address de tous les appareils dans le réseau	Type: Array of Address_Translation_Row Classification: Dynamic Accessibilité: Read only	Attribut structuré utilisé pour consulter les conversions d'adresses; voir Tableau 14
Réservé pour les éditions futures de la présente norme	2..63	-	-	-

La structure de données Address_Translation_Row est définie dans le Tableau 14.

Tableau 14 – Structure de données Address_Translation_Row

Nom du type de données normalisé: Address_Translation_Row		
Code du type de données normalisé: 402		
Nom de l'élément	**Identificateur de l'élément**	**Type de l'élément**
EUI64Address	1	Adresse EUI64Address de l'appareil unique au niveau global; Type: EUI64Address Classification: Static Accessibilité: Read only
IPv6Address	2	Adresse IPv6Address de l'appareil assignée par le gestionnaire de système; Type: IPv6Address Classification: Static Accessibilité: Read only
DL_Subnet_ID	3	Sous-réseau D dans lequel ou à partir duquel cet appareil est accessible; un appareil peut être accessible à partir de plusieurs sous-réseaux D, auquel cas cet élément correspond à un tel sous-réseau D; Type: Unsigned16 Classification: Static Accessibilité: Read Only
DL16Address	4	Adresse DL16Address de l'appareil dans le sous-réseau D indiquée par l'élément DL_Subnet_ID donné ci-dessus; Type: Unsigned16 Classification: Static Accessibilité: Read Only

Le service de consultation de conversions d'adresses est fourni par la méthode Read_Address_Row définie dans le Tableau 15.

Tableau 15 – Méthode Read_Address_Row

Nom du type d'objet normalisé: Directory service object (DSO, objet de service d'annuaire)				
Identificateur du type d'objet normalisé: 101				
Nom de la méthode	**ID de la méthode**	**Description de la méthode**		
Read_Address_Row	1	Méthode permettant d'utiliser le service de consultation de conversions d'adresses pour lire les valeurs d'autres adresses/identificateurs d'un appareil ayant reçu un indice (l'un de ses identificateurs ou l'une de ses adresses)		
	Arguments d'entrée			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	Attribute_ID	Data type: Unsigned8	Valeur = 1 (attribut Address_Translation_Table du DSO)
	2	Index_Info	Data type: Unsigned8	Indices nommés: 0: seule l'adresse EUI64Address est fournie; 1: seule l'adresse IPv6Address est fournie; 2: l'adresse EUI64Address et l'ID de sous-réseau D sont fournis; 3: l'adresse IPv6Address et l'ID de sous-réseau D sont fournis; 4: l'ID de sous-réseau D et l'adresse DL16Address sont fournis. Voir Tableau 16
	3	Index_EUI64	Data type: EUI64Address	Valeur: Adresse EUI64Address de l'appareil pour lequel la consultation d'adresse est nécessaire
	4	Index_128_Bit_Address	Data type: IPv6Address	Valeur: Adresse IPv6Address de l'appareil pour lequel la consultation d'adresse est nécessaire
	5	Index_DL_Subnet_ID	Data type: Unsigned16	Valeur: ID de sous-réseau D de l'appareil pour lequel la consultation d'adresse est nécessaire
	6	Index_DL_address_16_Bit	Data type: DL16Address	Valeur: Adresse DL16Address de l'appareil pour lequel la consultation d'adresse est nécessaire

	Arguments de sortie			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	Value_Type	Data type: Unsigned8	Type d'informations fournies Valeurs nommées: 0: une seule rangée si la valeur de l'ID de sous-réseau D est fournie comme argument d'entrée; 1: toutes les rangées si la valeur de l'ID de sous-réseau D n'est pas fournie comme argument d'entrée (c'est-à-dire que toutes les rangées pour l'adresse EUI64Address ou IPv6Address donnée sont retournées). Voir Tableau 17
	2	Value_Size	Data type: Unsigned8	Nombre de rangées retournées
	3	Data_Value_1	Data type: Address_Translation_Row	Adresse EUI64Address, adresse IPv6Address, ID de sous-réseau D, adresse DL16Address de l'appareil
	2+*n*	Data_Value_*n*	Data type: Address_Translation_Row	Adresse EUI64Address, adresse IPv6Address, ID de sous-réseau D, adresse DL16Address de l'appareil

Certains des arguments d'entrée ne sont pas applicables si l'argument Index_Info a certaines valeurs et ne doivent donc pas être inclus dans la demande. L'utilisation des arguments d'entrée pour la méthode Read_Address_Row est décrite dans le Tableau 16.

Tableau 16 – Utilisation des arguments d'entrée pour la méthode Read_Address_Row

Argument d'entrée	Non applicable à la valeur Index_Info
Attribute_ID	-
Index_Info	-
Index_EUI64	1, 3, 4
Index_128_Bit_Address	0, 2, 4
Index_DL_Subnet_ID	0, 1
Index_DL_Address_16_Bit	0, 1, 2, 3

Certains des arguments de sortie ne sont pas applicables si l'argument Value_Type a certaines valeurs et ne doivent donc pas être inclus dans la réponse. L'utilisation des arguments de sortie pour la méthode Read_Address_Row est décrite dans le Tableau 17.

Tableau 17 – Utilisation des arguments de sortie pour la méthode Read_Address_Row

Argument de sortie	Non applicable à la valeur Value_Type
Value_Type	-
Value_Size	0
Data_Value_1	-
Data_Value_n	0

Les interfaces de côté haut sur le DSO pour supprimer des adresses ou pour modifier des adresses ne sont pas spécifiées dans la présente norme.

6.3.5.5 Gestion des adresses DL16Address de multidiffusion

Les adresses DL16Address de la forme 0x 100x xxxx xxxx xxxx doivent être réservées pour la multidiffusion, en respectant la convention établie par l'IETF RFC 4944.

La gestion d'adresses DL16Address de multidiffusion n'est pas spécifiée dans la présente norme. Des attributs complémentaires pour l'objet service d'annuaire peuvent être spécifiés par les fournisseurs qui prennent en charge la gestion d'adresses DL16Address de multidiffusion.

6.3.6 Mise à niveau du firmware

Le gestionnaire de système fournit aux appareils la prise en charge pour les mises à niveau par liaison radio des firmwares. Le gestionnaire de système prend en charge les mises à jour de firmwares des suites de protocoles de communication.

Le gestionnaire de système doit fournir une interface pour accepter les mises à niveau de firmwares qui ont besoin d'être envoyées à tout appareil dans le réseau. Le gestionnaire de système doit utiliser l'UDO dans le DMAP de l'appareil pour envoyer cette mise à jour à l'appareil. Des mises à jour de firmwares des suites de protocoles de communication doivent être accomplies uniquement au moyen de l'UDO de gestionnaire de système. Cet UDO est décrit en 12.15.2.4.

Comme le gestionnaire de système maintient des informations relatives à tous les appareils dans le réseau, l'application de mise à jour d'hôte peut obtenir du gestionnaire de système des informations relatives aux appareils qui sont dans le réseau. L'application de mise à jour d'hôte peut utiliser ces informations pour déterminer quels appareils ont besoin de telles mises à niveau de firmware. La passerelle peut également être utilisée pour envoyer des mises à niveau de firmware à l'appareil. Si ces mises à niveau sont des mises à niveau de suites de protocoles de communication, elles doivent être envoyées au moyen de l'UDO de gestionnaire de système. Cela est décrit en U.3.2.

Le processus de mise à niveau de firmware doit assurer que les opérations de réseau sont maintenues à travers des mises à jour. Le processus peut inclure un mécanisme de basculement qui spécifie l'heure de basculement (après livraison de la mise à jour), le point auquel les appareils doivent commencer à utiliser le nouveau firmware. L'heure de basculement doit utiliser le sens partagé de temps qui a été configuré par le gestionnaire de système. Si le gestionnaire de système envoie la mise à niveau de firmware à l'appareil, il peut envoyer l'heure de basculement avec le téléchargement, ou il peut envoyer l'heure de basculement après que le téléchargement s'est achevé.

Comme les mises à niveau de firmware peuvent être spécifiques à un fournisseur, les mises à jour sont opaques pour le gestionnaire de système fournissant le service de mise à jour. Le gestionnaire de système accepte des mises à jour par l'intermédiaire des protocoles non spécifiés (un outil d'interface utilisateur avec un lecteur de DVD, par exemple) et fournit la mise à jour à des appareils choisis à des instants spécifiés en fonction de l'utilisateur ou autre entrée. Les détails relatifs aux questions de savoir comment l'application de mise à jour d'hôte communique avec le gestionnaire de système (telles que: quels appareils convient-il de mettre à jour?, quels sont leurs ID de fournisseur?, quand convient-il de mettre à jour les appareils?, et dans quel ordre convient-il de les mettre à jour?), ne sont pas spécifiés par la présente norme, mais de telles fonctions sont censées être prises en charge par le gestionnaire de système. Le gestionnaire de système peut programmer des mises à jour des appareils de façon à éviter ou réduire au maximum le temps d'interruption du réseau. Ce programme peut dépendre de la topologie du réseau.

La multidiffusion des mises à niveau de firmware n'est pas spécifiée par la présente norme.

L'UDO dans le gestionnaire de système doit être utilisé si le firmware du gestionnaire de système lui-même a besoin d'être mis à niveau. Les attributs, les méthodes et les

diagrammes d'état de l'UDO dans le gestionnaire de système sont selon la définition fournie en 12.15.2.4. L'identificateur d'objet pour l'UDO dans le SMAP doit être 7.

6.3.7 Surveillance des performances du système

6.3.7.1 Généralités

La surveillance des performances du système est effectuée par le gestionnaire de système afin de recueillir les informations qui peuvent être utilisées dans la prise des actions nécessaires pour optimiser les performances du système et pour réagir aux changements apportés à l'environnement radio et au statut de l'appareil. De telles actions sont effectuées par l'intermédiaire de la configuration de communication du système qui est décrite en 6.3.11.

La surveillance des performances du système est accomplie par l'intermédiaire du sondage des attributs d'appareil ou par la configuration des appareils pour générer des alertes qui fournissent des informations événementielles.

La surveillance des performances du système utilisant la publication périodique de rapports de santé issus des appareils est prise en charge par l'intermédiaire de l'utilisation du HRCO dans le DMAP de chaque appareil. Le HRCO, décrit en 6.2.7.7, peut être configuré par le gestionnaire de système pour rapporter périodiquement les valeurs d'un ou plusieurs attributs dans les objets de gestion de l'appareil. Avant que le gestionnaire de système ne configure le HRCO d'un quelconque appareil particulier pour éditer des rapports de santé, il a besoin de créer un objet de dispersion unique dans le SMAP pour agir comme étant l'abonné aux données qui seront éditées par le HRCO de cet appareil particulier. Des informations relatives à cet objet de dispersion unique doivent être acheminées vers le HRCO de cet appareil particulier en configurant l'attribut CommunicationEndPoint dans le HRCO. L'objet de dispersion est décrit en 12.15.2.6.

Des informations relatives aux capacités d'un nouvel appareil sont fournies au gestionnaire de système au cours de son processus de rattachement. Cela est débattu en 6.3.9.

Des appareils peuvent être configurés par le gestionnaire de système pour générer des alertes pour fournir des informations événementielles, par exemple, lorsqu'une liaison cesse de fonctionner ou lorsque la batterie d'un appareil de terrain a moins de 25 % de capacité restante. Cela est décrit en 6.3.7.2.

Alors que l'appareil mettant en œuvre le gestionnaire de système peut avoir une interface qui permet au personnel d'exploitation et de maintenance d'installation d'observer et commander les performances du réseau et des appareils, cette interface n'est ni obligatoire ni spécifiée par la présente norme.

L'UDO dans le DMAP d'un appareil peut être utilisé pour un téléchargement descendant, de l'appareil vers le gestionnaire de système, de grands blocs de données de performances d'appareil. De telles données peuvent être spécifiques à un fournisseur, spécifiques à un modèle ou spécifiques à une instance d'appareil. L'UDO dans le gestionnaire de système peut être utilisé pour un téléchargement montant de telles données en vue d'analyses complémentaires. Cette collecte de données et cette analyse des données ne sont pas spécifiées par la présente norme.

6.3.7.2 Alertes de la gestion de système

Le gestionnaire de système contient un objet récepteur d'alerte (ARO) qui reçoit des alertes de diagnostics de communication et des alertes de sécurité. De telles alertes sont décrites en 6.2.7.2. Ces alertes peuvent être utilisées par le gestionnaire de système pour surveiller les performances du système et entreprendre l'action adaptée, le cas échéant. L'identificateur d'objet pour l'ARO dans le SMAP doit être 8.

Après qu'un appareil a rejoint le réseau, il doit positionner les attributs Alert_Master_Comm_Diagnostics et Alert_Master_Security de l'ARMO afin qu'ils pointent vers le gestionnaire de système.

Les attributs de l'ARO dans le gestionnaire de système sont conformes à la définition donnée en 12.15.2.3. La valeur par défaut pour l'attribut ARO.Categories doit être 0110 0000.

Le diagramme d'états décrivant le traitement des rapports d'alertes par l'ARO est donné en 12.15.2.3.

6.3.7.3 System monitoring object (objet de surveillance de système)

Les attributs du SMO sont donnés dans le Tableau 18.

Tableau 18 – Attributs de SMO dans le gestionnaire de système

Nom du type d'objet normalisé: System monitoring object (SMO, objet de surveillance de système)				
Identificateur du type d'objet normalisé: 104				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Réservé pour les éditions futures de la présente norme	1..63	-	-	-

6.3.7.4 Configuration de la surveillance de système

La surveillance de performances de système peut avoir sa propre configuration permettant d'augmenter la surveillance lorsque cela est nécessaire. Par exemple, la liste d'alertes actives peut être réajustée en fonction de l'état courant du réseau. Pour le diagnostic du réseau, si un mode de défaillance est suspecté, le fait d'activer des alertes spécifiques peut fournir la preuve confirmant la défaillance en question. Une telle configuration du gestionnaire de système n'est pas spécifiée par la présente norme.

6.3.8 Service de configuration d'appareil

Avant qu'un appareil ne rejoigne le réseau, il exige les authentifiants de sécurité et les informations spécifiques au réseau qui sont appropriés. Ceux-ci sont fournis à l'appareil pendant le processus de configuration. Le processus de configuration, le rôle du gestionnaire de système dans ce processus et la définition de l'objet service de configuration d'appareil (DPO) sont décrits à l'Article 13.

6.3.9 Services de gestion d'appareil

6.3.9.1 Généralités

Un appareil conforme à la présente norme peut passer par plusieurs phases dans sa durée de vie opérationnelle. Ces phases sont décrites en 5.2.8. La gestion d'un appareil à travers certaines de ces phases est accomplie par le gestionnaire de système. Spécifiquement, le gestionnaire de système joue un rôle dans les processus de rattachement et de départ ainsi que dans la configuration de communication d'un appareil.

6.3.9.2 Processus de rattachement

6.3.9.2.1 Généralités

Un nouvel appareil doit obtenir de l'appareil de configuration les informations nécessaires relatives à la configuration. Cela est décrit à l'Article 13. L'attribut Join_Command dans le DMO d'un appareil doit être utilisé pour ordonner à l'appareil de rejoindre le réseau. Seul

l'appareil de configuration peut mettre à 1 l'attribut Join_Command, ce qui déclenche explicitement le processus de rattachement de l'appareil, étant donné que l'appareil n'a aucune connectivité à d'autres entités dans le réseau. L'attribut Join_Command ne doit être mis à 1 par l'appareil de configuration qu'à la suite d'une configuration réussie de l'appareil. Des routeurs d'annonce doivent relayer comme proxy les demandes de rattachement à la fréquence indiquée par l'attribut Proxy_Join_Request_Rate (attribut 36) du DMO. Cette fréquence assure que le réseau est protégé des attaques de déni de service tentées par l'intermédiaire des routeurs proxy. Pour une description de Proxy_Join_Request_Rate, voir le Tableau 10.

Le gestionnaire de système commande le traitement de nouveaux appareils rejoignant le réseau. Les appareils non rattachés qui mettent en œuvre une DLE selon la présente norme se mettent à l'écoute pour détecter des messages d'annonce issus de routeurs locaux dont les fonctions d'annonce sont configurées par le gestionnaire de système. Un tel routeur d'annonce doit apporter une assistance pendant le processus de rattachement d'un nouvel appareil en agissant comme un proxy pour le gestionnaire de système. Ce routeur d'annonce transmet la demande de rattachement issue du nouvel appareil vers le gestionnaire de système et transmet la réponse de rattachement issue du gestionnaire de système vers le nouvel appareil. Une demande de rattachement issue du nouvel appareil doit être traitée par le gestionnaire de système. Le gestionnaire de système doit générer une réponse de rattachement après avoir communiqué avec le gestionnaire de sécurité. Des routeurs d'annonce doivent relayer comme proxy les demandes de rattachement à la fréquence indiquée par l'attribut Proxy_Join_Request_Rate (attribut 36) du DMO. Cette fréquence protège le réseau contre des attaques de déni de service tentées par des routeurs proxy. Pour une description de cet attribut, voir le Tableau 10.

La demande de rattachement issue d'un nouvel appareil doit inclure des informations autres que de sécurité telles que l'adresse EUI64Address et les capacités de l'appareil ainsi que des informations de sécurité de l'appareil. La réponse de rattachement issue du gestionnaire de système doit inclure des informations autres que de sécurité telles que l'adresse IPv6Address assignée, l'adresse DL16Address assignée et des informations de contrat de l'appareil ainsi que des informations de sécurité telles que la clé T. Les informations de sécurité dans la demande de rattachement et dans la réponse de rattachement sont décrites en 7.4. Les contrats sont décrits en 6.3.11.2.

Plus de détails relatifs au processus de rattachement sont décrits en 7.4.

6.3.9.2.2 Méthodes de l'objet de gestion d'appareil pour routeur d'annonce

Le nouvel appareil doit utiliser la méthode Proxy_System_Manager_Join et la méthode Proxy_System_Manager_Contract définies pour le DMO du routeur d'annonce pour envoyer ses informations autres que de sécurité qui font partie intégrante de la demande de rattachement et pour obtenir ses informations autres que de sécurité qui font partie intégrante de la réponse de rattachement. Les informations autres que de sécurité qui font partie intégrante de la demande de rattachement sont réparties dans la demande de rattachement au réseau et la demande de contrat. Les informations autres que de sécurité qui font partie intégrante de la réponse de rattachement sont réparties dans la réponse de rattachement au réseau et la réponse de contrat. Les contrats sont décrits en 6.3.11.2.

La méthode Proxy_System_Manager_Join est définie dans le Tableau 19. La méthode Proxy_System_Manager_Contract est définie dans le Tableau 20.

Le nouvel appareil doit utiliser les méthodes définies en 7.4 pour le DMO du routeur d'annonce pour envoyer ses informations de sécurité qui font partie intégrante de la demande de rattachement et pour obtenir ses informations de sécurité qui font partie intégrante de la réponse de rattachement. L'utilisation de toutes ces méthodes par le nouvel appareil est décrite en 7.4.

L'accès au DMAP de l'appareil est limité au SMAP présent dans le gestionnaire de système. Le rattachement doit être autorisé seulement pour accéder aux méthodes

Proxy_System_Manager_Join et Proxy_System_Manager_Contract du DMO au cours du processus de rattachement.

Tableau 19 – Méthode Proxy_System_Manager_Join

<table>
<tr><th colspan="5">Nom du type d'objet normalisé: Device management object (DMO, objet de gestion d'appareil)</th></tr>
<tr><th colspan="5">Identificateur du type d'objet normalisé: 127</th></tr>
<tr><th>Nom de la méthode</th><th>ID de la méthode</th><th colspan="3">Description de la méthode</th></tr>
<tr><td>Proxy_System_Manager_Join</td><td>3</td><td colspan="3">Méthode permettant d'utiliser le routeur d'annonce comme gestionnaire de système proxy pour envoyer la demande de rattachement au réseau d'un nouvel appareil et obtenir la réponse de rattachement au réseau</td></tr>
<tr><td rowspan="12"></td><td colspan="4">Arguments d'entrée</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td>1</td><td>EUI64</td><td>EUI64Address</td><td>Attribut DMO EUI64; voir Tableau 10</td></tr>
<tr><td>2</td><td>DL_Subnet_ID</td><td>Unsigned16</td><td>Sous-réseau D que le nouvel appareil essaie de rejoindre; il s'agit également du sous-réseau D du routeur d'annonce.
Valeur de données:
0: l'appareil ne fait partie d'aucun sous-réseau D</td></tr>
<tr><td>3</td><td>Device_Role_Capability</td><td>Unsigned16</td><td>Attribut DMO Device_Role_Capability; voir Tableau 10</td></tr>
<tr><td>4</td><td>Size_of_Tag_Name</td><td>Type: Unsigned8</td><td>Longueur en octets de Tag_Name</td></tr>
<tr><td>5</td><td>Tag_Name</td><td>Type: VisibleString SIZE(0..16)</td><td>Attribut DMO Tag_Name; voir Tableau 10</td></tr>
<tr><td>6</td><td>Comm_SW_Major_Version</td><td>Type: Unsigned8</td><td>Attribut DMO Comm_SW_Major_Version; voir Tableau 10</td></tr>
<tr><td>7</td><td>Comm_SW_Minor_Version</td><td>Type: Unsigned8</td><td>Attribut DMO Comm_SW_Minor_Version; voir Tableau 10</td></tr>
<tr><td>8</td><td>Size_of_Software_Revision_Information</td><td>Type: Unsigned8</td><td>Taille en octets de Software_Revision_Information</td></tr>
<tr><td>9</td><td>Software_Revision_Information</td><td>Type: VisibleString SIZE(0..16)</td><td>Attribut DMO Software_Revision_Information; voir Tableau 10</td></tr>
<tr><td>10</td><td>DeviceCapability</td><td>Type: OctetString</td><td>Attribut DLMO DeviceCapability; voir Tableau 141</td></tr>
</table>

Nom du type d'objet normalisé: Device management object (DMO, objet de gestion d'appareil)				
Identificateur du type d'objet normalisé: 127				
Nom de la méthode	**ID de la méthode**	**Description de la méthode**		
	Arguments de sortie			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	Assigned_Network_ Address_128_Bit	Type: IPv6Address	Cette valeur est écrite dans l'attribut DMO Network_Address_128_Bit; voir Tableau 10
	2	Assigned_DL_ Address_16_Bit	Type: DL16Address	Cette valeur est écrite dans l'attribut DMO DL_Address_16_Bit; voir Tableau 10
	3	Assigned_Device_Rol e	Type: BitArray16	Cette valeur est écrite dans l'attribut DMO Assigned_Device_Role; voir Tableau 10
	4	System_Manager_ Network_Address_ 128_Bit	Type: IPv6Address	Cette valeur est écrite dans l'attribut DMO System_Manager_128_Bit_Address; voir Tableau 10
	5	System_Manager_DL _ Address_16_Bit	Type: DL16Address	Cette valeur est écrite dans l'attribut DMO System_Manager_DL_Address_16_ Bit; voir Tableau 10
	6	System_Manager_ EUI64	Type: EUI64Address	Cette valeur est écrite dans l'attribut DMO System_Manager_EUI64Address; voir Tableau 10
	7	MIC	Type: OctetString4	Cette valeur est utilisée pour protéger les arguments 1 à 6 avec la clé de rattachement. Cette valeur MIC est générée par le gestionnaire de sécurité. Le routeur d'annonce ne doit pas écraser en écriture cette valeur. Voir 7.4.4.3.2
	8	Assigned_Max_TSDU _Size	Type: Unsigned16	Indique la TSDU maximale prise en charge en octets qui peut être convertie par la source en taille d'APDU maximale en prenant en compte les tailles de la TL, de la sécurité, des en-têtes d'AL et du TMIC

Tableau 20 – Méthode Proxy_System_Manager_Contract

<table>
<tr><th colspan="5">Nom du type d'objet normalisé: Device management object (DMO, objet de gestion d'appareil)</th></tr>
<tr><th colspan="5">Identificateur du type d'objet normalisé: 127</th></tr>
<tr><th>Nom de la méthode</th><th>ID de la méthode</th><th colspan="3">Description de la méthode</th></tr>
<tr><td rowspan="8">Proxy_System_ Manager_Contract</td><td>4</td><td colspan="3">Méthode permettant d'utiliser le routeur d'annonce comme gestionnaire de système proxy pour envoyer la demande de contrat d'un nouvel appareil et obtenir la réponse de contrat. Les contrats sont décrits en 6.3.11.2.</td></tr>
<tr><td colspan="4">Arguments d'entrée</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td>1</td><td>EUI64</td><td>Type: EUI64Address</td><td>Attribut DMO EUI64; voir Tableau 10</td></tr>
<tr><td colspan="4">Arguments de sortie</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td>1</td><td>Contract_Response</td><td>Type: New_Device_Contract_Response (voir Tableau 31)</td><td>Réponse de contrat pour prendre en charge une future communication du nouvel appareil vers le gestionnaire de système; les contrats sont décrits en 6.3.11.2.</td></tr>
<tr><td>2</td><td>MIC</td><td>Type: OctetString4</td><td>Cette valeur est utilisée pour protéger l'argument 1 avec la clé de rattachement Cette valeur MIC est générée dans le gestionnaire de sécurité. Le routeur d'annonce ne doit pas écraser en écriture cette valeur.</td></tr>
</table>

6.3.9.2.3 Capacités de nouvel appareil

Les informations relatives aux capacités d'un nouvel appareil en ce qui concerne le rôle de l'appareil doivent être fournies au gestionnaire de système au cours du processus de rattachement de l'appareil. Ces informations sont décrites dans le Tableau 19.

6.3.9.3 Configuration d'appareil

Un appareil est configuré après avoir rejoint un réseau. La configuration d'appareil inclut d'obtenir des ressources de communication pour prendre en charge les besoins de communication de l'appareil et de configurer la pile de protocoles de l'appareil pour utiliser ces ressources pour la communication. Pendant cette configuration, le gestionnaire de système peut tenir compte des capacités d'un appareil. Un appareil peut être reconfiguré lorsque le réseau change ou lorsque les applications sur l'appareil ont besoin de changer leurs services.

La configuration d'appareil est habituellement accomplie pendant l'établissement de contrats. Cela est décrit en 6.3.11.2. Les attributs et les méthodes définis pour les objets de gestion du DMAP doivent être utilisés par le gestionnaire de système pour configurer l'appareil.

Le gestionnaire de système ne configure pas les UAP sur l'appareil. Cela est fait par des applications d'hôte sur des réseaux d'installation ou par des outils de maintenance tenus à la main.

6.3.9.4 Processus de départ

6.3.9.4.1 Généralités

Le gestionnaire de système commande le processus consistant pour un appareil précédemment rattaché à quitter le réseau. Ce processus de départ peut être initié par l'appareil lorsqu'il a l'intention de quitter le réseau ou il peut être initié par le gestionnaire de système.

Le processus de départ inclut deux scénarios: le redémarrage de l'appareil et la réinitialisation de l'appareil aux valeurs d'usine par défaut.

Le redémarrage d'appareil se produit lorsque l'appareil lui-même ou le gestionnaire de système amène l'appareil à redémarrer. Quelques exemples qui conduisent à ce scénario sont le remplacement de batterie ou le réamorçage pour appliquer une nouvelle image de firmware. Un appareil est réinitialisé à ses valeurs de réglage d'usine par défaut si l'appareil retourne à son état d'usine par défaut. Quelques exemples qui conduisent à ce scénario sont le fait que l'appareil est remis dans le stock général en vue d'un futur déploiement ou que l'appareil est déplacé vers un réseau différent.

6.3.9.4.2 Redémarrage d'appareil

Un processus de redémarrage d'appareil peut être initié par l'appareil lui-même ou par le gestionnaire de système. Il y a deux types de redémarrages: warmRestart et restartAsProvisioned. Dans tous les deux cas, les appareils déclencheront immédiatement le processus de rattachement à la suite de l'événement de redémarrage. Une écriture explicite de l'attribut Join_Command DMO à 1 n'est pas nécessaire à la suite d'un événement warmRestart ou restartAsProvisioned.

L'attribut Join_Command dans le DMO d'un appareil doit être utilisé pour ordonner à l'appareil d'accomplir soit un warmRestart, soit un restartAsProvisioned.

Un appareil qui reçoit la commande de warmRestart doit se réamorcer lui-même. Si l'attribut Non_Volatile_Memory_Capability dans le DMO est 1, l'appareil doit retenir les valeurs de tous les attributs constants et statiques dans tous les objets d'application présents dans le DMAP ainsi que dans les UAP de l'appareil. Tous les autres attributs sont réinitialisés à leurs valeurs par défaut. Si l'attribut Non_Volatile_Memory_Capability dans le DMO est 0, l'appareil doit retenir toutes les informations qui lui ont été fournies au cours de l'étape de configuration avant qu'il n'ait rejoint le réseau pour la première fois ainsi que toutes les informations constantes et statiques présentes dans les UAP. Tous les autres attributs sont réinitialisés à leurs valeurs par défaut et l'appareil retourne à son état configuré.

Un appareil qui reçoit la commande restartAsProvisioned doit réinitialiser tous les attributs constants et statiques dans tous les objets d'application présents dans le DMAP indépendamment de la valeur de réglage de l'attribut Non_Volatile_Memory_Capability présente dans le DMO, à l'exception du DPO. Un appareil qui reçoit la commande restartAsProvisioned doit se réamorcer lui-même tout en retenant toutes les informations qui lui avaient été fournies au cours de l'étape de configuration avant qu'il n'ait rejoint le réseau pour la première fois. Ces informations sont décrites à l'Article 13. Ces informations sont habituellement nécessaires à l'appareil pour rejoindre le réseau sans devoir passer une nouvelle fois par l'étape de configuration. L'appareil doit également retenir toutes les informations constantes et statiques présentes dans les UAP.

Le Tableau 21 recueille et présente les effets des différentes commandes de rattachement sur divers ensembles d'attributs.

Tableau 21 – Effet des différentes commandes de rattachement sur des ensembles d'attributs

Type de commande de rattachement	Attributs de DMAP (à l'exception du DPO)	Attributs d'UAP	Attributs de DPO
WarmRestart (Join_Command =2) when Non_Volatile_Memory_Capability = 1	KEEP	KEEP	KEEP
WarmRestart (Join_Command =2) when Non_Volatile_Memory_Capability = 0	CLEAR	KEEP	KEEP
RestartAsProvisioned (Join_Command = 3)	CLEAR	KEEP	KEEP
Réinitialisation aux valeurs d'usine par défaut (Join_Command = 4)	CLEAR	CLEAR	CLEAR

Un téléchargement descendant de firmware peut conduire à un redémarrage d'appareil. Les informations nécessaires à l'appareil pour utiliser le nouveau firmware et rejoindre le réseau peuvent être stockées dans l'appareil. L'appareil passe habituellement par un cycle de restartAsProvisioned dans ces cas.

6.3.9.4.3 Appareil réinitialisé aux valeurs d'usine par défaut

Un processus de réinitialisation d'appareil peut être initié par l'appareil lui-même ou par le gestionnaire de système.

L'attribut Join_Command dans le DMO d'un appareil doit être utilisé pour ordonner à l'appareil d'accomplir une réinitialisation. Une commande de réinitialisation force l'appareil à se réinitialiser aux valeurs d'usine par défaut, et tous les attributs sont réinitialisés à leurs valeurs par défaut. L'appareil est censé retourner à l'état d'usine par défaut qui est décrit dans l'Article 13.

6.3.9.4.4 Remplacement d'appareil

Si un ancien appareil (à savoir un appareil rattaché) est remplacé par un nouvel appareil (à savoir un appareil non rattaché), le gestionnaire de système est censé fournir l'ancienne adresse IPv6Address à cet appareil de remplacement. L'application d'hôte ou l'utilisateur est censé(e) informer le gestionnaire de système au sujet de ce remplacement à travers le chemin de communication 5 à la Figure 23. Le gestionnaire de système peut choisir de configurer d'autres attributs dans le DMAP de l'appareil de remplacement pour correspondre à ceux contenus dans l'ancien appareil. La configuration des UAP dans l'appareil de remplacement est censée être effectuée par des applications d'hôtes sur des réseaux d'installation ou par des outils de maintenance tenus à la main.

6.3.9.5 Device management service object (objet de service de gestion d'appareil)

L'objet de service de gestion d'appareil (DMSO) dans le gestionnaire de système doit traiter les informations autres que de sécurité dans la demande de rattachement issue du nouvel appareil qui sont transmises par le routeur d'annonce et doit générer des informations autres que de sécurité dans la réponse de rattachement.

Les attributs du DMSO sont donnés dans le Tableau 22.

Tableau 22 – Attributs du DMSO dans le gestionnaire de système

Nom du type d'objet normalisé: Device management service object (DMSO, objet de gestion de sécurité d'appareil)				
Identificateur du type d'objet normalisé: 103				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Réservé pour les éditions futures de la présente norme	1..63	-	-	-

Un nouvel appareil communique avec un routeur d'annonce qui agit comme un proxy pour le gestionnaire de système et transmet tous les messages de rattachement entre le nouvel appareil et le gestionnaire de système. Le processus de rattachement est décrit en 7.4. Les méthodes utilisées pour envoyer des messages de demande de rattachement et de réponse de rattachement entre le nouvel appareil et le routeur d'annonce sont données en 6.3.9.2.2. Le routeur d'annonce doit utiliser la méthode System_Manager_Join et la méthode System_Manager_Contract définies dans le DMSO pour envoyer la demande de rattachement au réseau et la demande de contrat et pour recevoir la réponse de rattachement au réseau et la réponse de contrat qui sont associées au processus de rattachement de ce nouvel appareil.

L'objet source des méthodes System_Manager_Join et System_Manager_Contract est le DMO routeur proxy d'annonce qui communique avec le gestionnaire de système pour le compte du nouvel appareil.

La méthode System_Manager_Join est définie dans le Tableau 23. La méthode System_Manager_Contract est définie dans le Tableau 24.

Tableau 23 – Méthode System_Manager_Join

Nom du type d'objet normalisé: Device management service object (DMSO, objet de gestion de sécurité d'appareil)				
Identificateur du type d'objet normalisé: 103				
Nom de la méthode	**ID de la méthode**	**Description de la méthode**		
System_Manager_Join	1	Méthode permettant d'envoyer au gestionnaire de système la demande de rattachement au réseau d'un nouvel appareil et d'obtenir la réponse de rattachement au réseau		
	Arguments d'entrée			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	EUI64	Type: EUI64Address	Attribut DMO EUI64; voir Tableau 10
	2	DL_Subnet_ID	Type: Unsigned16	Sous-réseau D que le nouvel appareil essaie de rejoindre; il s'agit également du sous-réseau D du routeur d'annonce. Valeurs nommées: 0: l'appareil ne fait partie d'aucun sous-réseau D
	3	Device_Role_Capability	Type: Unsigned16	Attribut DMO Device_Role_Capability; voir Tableau 10

Nom du type d'objet normalisé: Device management service object (DMSO, objet de gestion de sécurité d'appareil)				
Identificateur du type d'objet normalisé: 103				
Nom de la méthode	**ID de la méthode**	**Description de la méthode**		
	4	Size_of_Tag_Name	Type: Unsigned8	Longueur en octets de Tag_Name
	5	Tag_Name	Type: VisibleString SIZE(0..16)	Attribut DMO Tag_Name; voir Tableau 10
	6	Comm_SW_Major_Version	Type: Unsigned8	Attribut DMO Comm_ SW_Major_Version; voir Tableau 10
	7	Comm_SW_Minor_Version	Type: Unsigned8	Attribut DMO Comm_ SW_Minor_Version; voir Tableau 10
	8	Size_of_Software_ Revision_Information	Type: Unsigned8	Longueur en octets de Software_Revision_ Information
	9	Software_Revision_Information	Type: VisibleString SIZE(0..16)	Attribut DMO Software_Revision_ Information; voir Tableau 10
	10	DeviceCapability	Type: OctetString	Attribut DLMO DeviceCapability; voir Tableau 141

	Arguments de sortie			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	Assigned_Network_ Address_128_Bit	Type: IPv6Address	Cette valeur est écrite dans l'attribut DMO Network_Address_ 128_Bit; voir Tableau 10
	2	Assigned_DL_Address_16_Bit	Type: DL16Address	Cette valeur est écrite dans l'attribut DMO DL_Address_16_Bit; voir Tableau 10
	3	Assigned_Device_Role	Type: BitArray16	Cette valeur est écrite dans l'attribut DMO Assigned_Device_Role; voir Tableau 10
	4	System_Manager_ Network_Address_128_Bit	Type: IPv6Address	Cette valeur est écrite dans l'attribut DMO System_Manager_ 128_Bit_Address; voir Tableau 10
	5	System_Manager_ DL_Address_16_Bit	Type: DL16Address	Cette valeur est écrite dans l'attribut DMO System_Manager_ DL_Address_16_Bit; voir Tableau 10
	6	System_Manager_EUI64	Type: EUI64Address	Cette valeur est écrite dans l'attribut DMO System_Manager_EUI64Address; voir Tableau 10

Nom du type d'objet normalisé: Device management service object (DMSO, objet de gestion de sécurité d'appareil)				
Identificateur du type d'objet normalisé: 103				
Nom de la méthode	**ID de la méthode**	**Description de la méthode**		
	7	MIC	Type: OctetString4	Cette valeur est utilisée pour protéger les arguments 1 à 6. Cette valeur MIC est générée par le gestionnaire de sécurité. Voir 7.4.4.3.2
	8	Assigned_Max_TSDU_Size	Type: Unsigned16	Indique la TSDU maximale prise en charge en octets qui peut être convertie par la source en taille d'APDU maximale en prenant en compte les tailles de la TL, de la sécurité, des en-têtes d'AL et du TMIC.

Tableau 24 – Méthode System_Manager_Contract

Nom du type d'objet normalisé: Device management service object (DMSO, objet de gestion de sécurité d'appareil)				
Identificateur du type d'objet normalisé: 103				
Nom de la méthode	**ID de la méthode**	**Description de la méthode**		
System_Manager_Contract	2	Méthode permettant d'envoyer au gestionnaire de système la demande de contrat d'un nouvel appareil et d'obtenir la réponse de contrat. Les contrats sont décrits en 6.3.11.2.		
	Arguments d'entrée			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	EUI64	Type: EUI64Address	Adresse EUI64Address du nouvel appareil tentant de rejoindre le réseau
	Arguments de sortie			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	Contract_Response	Type: New_Device_Contract_Response (voir Tableau 31)	Réponse de contrat pour prendre en charge une future communication du nouvel appareil vers le gestionnaire de système; les contrats sont décrits en 6.3.11.2
	2	MIC	Type: OctetString4	Cette valeur est utilisée pour protéger les arguments 1 à 6 avec la clé de rattachement. Cette valeur MIC est générée dans le gestionnaire de sécurité. Voir 7.4.4.3.2

Lorsque le DMSO génère des réponses de System_Manager_Join et de System_Manager_Contract, il envoie d'abord ces réponses au PSMO qui les envoie à son

tour au gestionnaire de sécurité. Le gestionnaire de sécurité doit protéger ces réponses avec un champ MIC en utilisant la clé de rattachement et les retourner au PSMO qui, à son tour, les remet en retour au DMSO. Le DMSO doit alors envoyer les réponses en retour au routeur d'annonce. Cette interaction avec le PSMO est décrite en 7.4.

6.3.10 Services de temps système

6.3.10.1 Généralités

Le temps dans la présente norme est basé sur le temps atomique international (TAI) comme référence temporelle. Le temps dans la présente norme est communiqué sous la forme de secondes écoulées depuis le 1er janvier 1958 (c'est-à-dire 1958/01/01 00:00).

Le système prend en charge la synchronisation afin que, au niveau de l'appareil, les applications puissent utiliser le temps pour coordonner des activités ou horodater des informations, améliorant l'utilisation de l'énergie et la fiabilité. Le temps système doit être disponible auprès d'au moins un appareil (une source de temps système) sur le réseau.

Une source de temps TAI n'est pas exigée pour l'exploitation d'un réseau conforme à la présente norme. Les sources alternatives de temps sont converties en unités TAI. La base de temps utilisée par le réseau doit être, à ±1 s près, le temps TAI réel. La passerelle doit convertir le temps TAI nominal du réseau en référence de temps système local s'il est disponible.

Le gestionnaire de système doit configurer au moins une source de temps système dans chaque sous-réseau D du réseau. Le gestionnaire de système lui-même peut être la source de temps système. Cela est décrit en 6.3.10.3. Le gestionnaire de système doit également configurer la topologie de distribution pour la diffusion du temps dans le sous-réseau D et pour la synchronisation des horloges d'appareil. Le gestionnaire de système configure chaque appareil dans le sous-réseau D avec l'/les horloge(s) parente(s) que l'appareil doit utiliser pour synchroniser son horloge. La DL dans chaque appareil est responsable de mesurer le temps et de garder les horloges synchronisées. Cela est décrit en 9.1.9.

Les routeurs dorsaux sont censés utiliser des techniques propriétaires ou normalisées pour maintenir la synchronisation. Ces techniques ne sont pas spécifiées par la présente norme.

Les appareils ayant besoin de convertir le temps TAI au format hh:mm:ss (tel que sur un écran utilisateur) peuvent rendre compte d'un réajustement de secondes intercalaires accumulées en temps universel coordonné (TUC).[6] Le gestionnaire de système doit fournir ce réajustement du TUC à ces appareils. Si l'appareil a besoin de ces informations de réajustement de TUC issues du gestionnaire de système, il convient qu'il les rafraîchisse peu souvent, mais périodiquement, comme au début de chaque mois ou à une quelconque autre limite arbitraire d'horloge.

Tous les appareils dans un réseau conforme à la présente norme partagent la référence de temps TAI avec des degrés variables d'exactitude. Pour prendre en charge une séquence d'événements ou autres opérations de temporisation des processus d'application, il convient que tous les appareils de routage au sein d'un réseau conforme à la présente norme soient exacts à ±10 ms près. L'exigence en matière de précision de l'horloge pour chaque appareil de routage est décrite en 9.1.9.2.2.

Le gestionnaire de système est responsable de coordonner le temps à travers différents sous-réseaux D en sélectionnant les sources appropriées de temps système dans chaque sous-réseau D. Cette coordination n'est pas spécifiée par la présente norme.

---

6 Une liste de tels réajustements est mise à jour à l'adresse ftp://maia.usno.navy.mil/ser7/tai-utc.dat.

Pour prendre en charge une séquence d'événements ou autres opérations de temporisation des processus d'application, il convient que les routeurs dorsaux soient également exacts à ±10 ms près. Ni la conversion des unités de temps utilisées par les routeurs dorsaux ni le réajustement pour s'aligner avec le temps TAI utilisé par les appareils conformes à la présente norme ne sont spécifiés par la présente norme.

Si le gestionnaire de système fait partie intégrante du sous-réseau D, alors la DL dans le gestionnaire de système est responsable de mesurer le temps et de garder l'horloge d'appareil synchronisée. Si le gestionnaire de système est connecté à la dorsale, il est censé maintenir la synchronisation d'horloge avec le reste du réseau et maintenir les informations relatives au temps en unités TAI. Dans ce cas, les techniques pour le faire ne sont pas spécifiées par la présente norme.

Les couches de protocoles qui requièrent le temps TAI courant de l'appareil peuvent l'obtenir auprès du DMO dans le DMAP.

6.3.10.2 Capacités d'exactitude d'horloge d'appareil

Le gestionnaire de système a besoin de connaître l'exactitude d'horloge de chaque appareil. Par exemple, il a besoin savoir si un appareil est capable de maintenir l'exactitude de ±1 ms pendant 30 s sans mise à jour d'horloge. Une telle information relative à un appareil doit être fournie au gestionnaire de système par le DLMO. Cela est décrit en Tableau 147.

6.3.10.3 Sélection de la source de temps système

L'appareil mettant en œuvre le rôle de gestionnaire de système peut également mettre en œuvre le rôle de source de temps système dans un réseau, ou il peut déléguer le rôle de source de temps système à n'importe quel(s) appareil(s) dans le réseau capable(s) de jouer ce rôle comme indiqué par l'attribut Device_Role_Capability dans le DMO de l'appareil. Le gestionnaire de système doit utiliser l'attribut Assigned_Device_Role dans le DMO de l'appareil pour le configurer comme source de temps système. Le gestionnaire de système peut sélectionner la source de temps système en fonction des capacités d'exactitude d'horloge de l'appareil.

La source de temps système est la source ultime du sens de temps dans un sous-réseau D. La source de système au sein d'un sous-réseau D doit être exacte à ±1 s près du temps TAI réel et doit augmenter de façon monotone à un taux qui poursuit le temps TAI avec une erreur maximale de 1×10^{-6}, c'est-à-dire que le taux d'accroissement du temps doit être exact, même si la source de temps elle-même est relativement inexacte.

Si plusieurs sources de temps système existent au sein d'un sous-réseau D, il convient qu'elles se poursuivent l'une l'autre à 0,1 ms près. Si la synchronisation de 0,1 ms parmi les sources de temps système du sous-réseau D ne peut pas être obtenue, le gestionnaire de système doit dicter l'instant où un appareil commute d'une source de temps système à l'autre. L'attribut dlmo.ClockStale décrit en 9.1.9.2.3 et en 9.4.2.14 doit être utilisé pour informer l'appareil de l'instant auquel commuter vers une autre source de temps système. Des chemins de propagation de temps sont décrits en 6.3.10.4. Si les appareils doivent commuter de sources de temps système, les chemins de propagation de temps peuvent être réarrangés par le gestionnaire de système selon ce qui est approprié.

Le gestionnaire de système doit s'assurer que chaque sous-réseau D a au moins une source de temps système. Quelques exemples de sources de temps système incluent:

- Dans une application extérieure, le gestionnaire de système peut désigner quelques appareils avec des capacités de système de positionnement global (GPS) comme sources de temps système. Le temps peut être propagé à travers les sous-réseaux D à partir de ces sources.
- Dans un grand réseau avec une dorsale Ethernet, la dorsale elle-même peut fournir un service de temps qui est synchronisé à 0,1 ms près à une référence de temps partagé

pour les appareils qui sont sur la dorsale. Le gestionnaire de système peut désigner les routeurs dorsaux comme sources de temps système, et le temps peut être propagé de ces routeurs dorsaux vers des appareils dans les sous-réseaux D.

- Le gestionnaire de système peut périodiquement synchroniser à une source de temps distante par l'intermédiaire d'une connexion, sans fil ou câblée, de longue distance. Cette source de temps fournit une exactitude de ±1 s. Le gestionnaire de système peut alors agir comme la source de temps système dans le réseau.

Si les plusieurs sources de temps système existent dans un sous-réseau D, le gestionnaire de système peut assigner une des sources de temps système comme étant la source par défaut et les autres comme étant des secours. Les techniques pour une telle assignation ne sont pas spécifiées par la présente norme.

Des corrections d'horloge au sein d'une source de temps système sont habituellement appliquées à un taux qui peut s'assurer que la correction n'excède pas 0,5 ms dans une période donnée de 30 s. Les corrections discontinues d'horloge sont prises en charge, avec l'ordre donné aux appareils sur un sous-réseau D de réajuster leurs horloges à un instant spécifique. Cela est décrit en 9.1.9.3.6.

6.3.10.4 Topologie de distribution du temps

En plus des sources de temps système, le gestionnaire de système configure également des destinataires d'horloges et des répétiteurs d'horloges dans un sous-réseau D.

Tous les appareils dans un sous-réseau D, à l'exception des sources de temps système, sont configurés comme destinataires d'horloge, c'est-à-dire qu'ils reçoivent les mises à jour périodiques d'horloge issues d'une ou plusieurs sources d'horloge dans leurs voisinages immédiats. Une source d'horloge peut être une source de temps système ou un répétiteur d'horloge.

Les répétiteurs d'horloge sont des destinataires d'horloge qui agissent également comme sources d'horloge pour certains voisins. Les répétiteurs d'horloge propagent le temps à travers un sous-réseau D. L'exigence en matière de précision de l'horloge pour un répétiteur d'horloge est décrite en 9.1.9.2.2.

Les relations source/destinataire d'horloge et, donc, les topologies de distribution du temps, sont définies par le gestionnaire de système. Les chemins de propagation du temps dans ces topologies correspondent habituellement à des graphes de cheminement, mais cela n'est exigé. Les chemins circulaires de propagation de temps ne sont pas autorisés. La propagation d'horloge peut être arrangée afin que les répétiteurs d'horloge fournissent des mises à jour à leurs destinataires peu après qu'ils ont reçu eux-mêmes leurs mises à jour.

Le gestionnaire de système utilise le DLMO d'un appareil pour configurer sa/ses source(s) d'horloge. Le temps d'un destinataire d'horloge peut être mis à jour au cours de chaque interaction avec une source d'horloge désignée. La sélection des sources d'horloge et la temporisation des mises à jour d'horloge sont arrangées par le gestionnaire de système. Ces mises à jour d'horloge sont décrites en 9.1.9.2.

6.3.10.5 Surveillance de l'exactitude de la synchronisation du temps

La gestion de système peut prendre en charge des mécanismes pour recueillir des informations relatives à l'exactitude du sens de temps distribué, ainsi que pour produire une alerte lorsque le sens du temps entre une paire d'appareils varie suffisamment au point de causer des problèmes au sein du système. Les alertes décrites en 9.6 peuvent être utilisées dans ce but par le gestionnaire de système.

Les attributs spécifiés par un fournisseur dans le DMO d'un appareil peuvent être utilisés pour recueillir de telles informations. Les alertes spécifiées par un fournisseur issues du DMO

peuvent être utilisées pour diagnostiquer des problèmes relatifs à la synchronisation d'horloge et à la maintenance d'horloge.

6.3.10.6 System time service object (objet de service de temps système)

Le gestionnaire de système contient l'objet de service de temps système (STSO), qui doit fournir l'ajustement de secondes intercalaires accumulées TUC aux appareils dans le réseau. D'autres attributs spécifiés par un fournisseur peuvent être ajoutés au STSO.

Les attributs du STSO dans le gestionnaire de système sont donnés dans le Tableau 25.

NOTE Pour plus d'informations sur cet ajustement de secondes intercalaires, voir https://en.wikipedia.org/wiki/Leap_second.

Tableau 25 – Attributs du STSO dans le gestionnaire de système

Nom du type d'objet normalisé: System time service object (STSO, objet de service de temps système)				
Identificateur du type d'objet normalisé: 100				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Current_UTC_Adjustment	1	Valeur courante de l'ajustement de secondes intercalaires accumulées du TUC	Type: Integer16 Classification: Dynamic Accessibilité: Read only Valeur par défaut: 35	Les appareils qui ont besoin de convertir le temps TAI au format hh:mm:ss ont besoin de ce réajustement issu du gestionnaire de système; unités en secondes; noter que l'ajustement peut être négatif; noter que le TUC et le TAI sont basés sur des dates de début différentes, mais cette différence n'est pas couverte par cet attribut; le 2012.06.30 23:59:60, la valeur est passée de 34 s à 35 s. Le mécanisme utilisé par le gestionnaire de système pour obtenir cet ajustement n'est pas spécifié.
Next_UTC_Adjustment_Time	2	Temps TAI lorsque la valeur du réajustement TUC changera par rapport à l'actuel	Type: TAITimeRounded Classification: Dynamic Accessibilité: Read only Valeur par défaut: Voir description	Si le gestionnaire de système connaît le prochain temps auquel cette valeur de réajustement TUC changera, le SM est censé indiquer ce temps en unités TAI. Si le gestionnaire de système ne connaît pas ce temps, il est censé indiquer le temps TAI courant et, en conséquence, la valeur de l'attribut Next_UTC_Adjustment doit être la même que la valeur de Current_UTC_Adjustment.
Next_UTC_Adjustment	3	Prochaine valeur de l'ajustement de secondes intercalaires accumulées du TUC	Type: Integer16 Classification: Dynamic Accessibilité: Read only Valeur par défaut: 35	Ajustement TUC qui entrera en vigueur au temps spécifié par l'attribut Next_UTC_Adjustment_Time
Réservé pour les éditions futures de la présente norme	4..63	-	-	-
NOTE Les temps TUC, TAI et GPS sont basés sur des dates de début différentes, mais cette différence n'est pas couverte par cet attribut. Le 2009/01/01, la valeur Current_UTC_Adjustment est passée de 33 s à 34 s. Le temps GPS est toujours en retard de 19 s par rapport au temps TAI. Les informations de temps GPS incluent le décalage nécessaire pour effectuer la conversion au format ou partir du format TUC.				

6.3.11 Configuration de communication de système

6.3.11.1 Généralités

Le gestionnaire de système fournit la commande de la configuration de communication de système en temps d'exécution. Il prend en charge la configuration du réseau, y compris les attributs de la suite de protocoles de la DL à l'AL.

La configuration de communication de système inclut l'assignation d'intervalles de temps, de modèles et de graphes aux appareils dans le réseau. Il convient que le gestionnaire de système prenne en compte les capacités de l'appareil dans le réseau tout en configurant de telles assignations. Le cas échéant, il convient que le système soit reconfiguré pour récupérer après des scénarios de défaillance.

6.3.11.2 Services de contrat

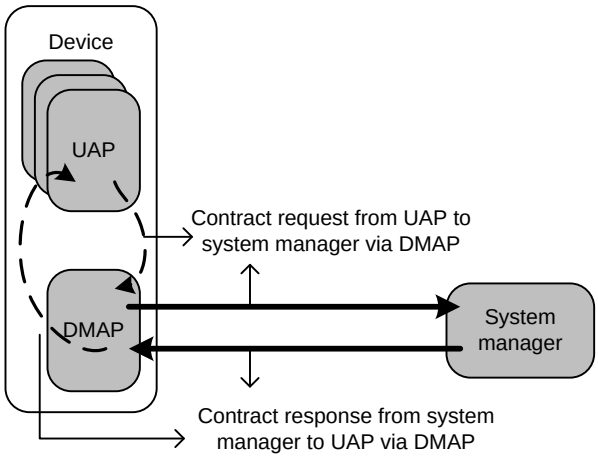
6.3.11.2.1 Définition de contrat

La configuration de communication de système est réalisée au moyen des services de contrat fournis par le gestionnaire de système.

Un contrat se réfère à un accord entre le gestionnaire de système et un appareil dans le réseau qui doit impliquer l'allocation de ressources du réseau par le gestionnaire de système pour prendre en charge un besoin particulier de communication de cet appareil. Cet appareil est la source des messages de communication et l'appareil avec lequel il souhaite communiquer est la destination.

Un contrat doit établir et prendre en charge le chemin de communication entre les appareils dans le réseau qui sont conformes à la présente norme pour prendre en charge le besoin de communication d'un processus d'application. Un processus d'application qui requiert la communication avec un processus d'application dans un autre appareil doit demander un contrat. De telles demandes de contrat peuvent provenir d'un processus d'application dans l'un quelconque des appareils conformes à la présente norme, tels qu'un appareil de terrain, une passerelle, un routeur dorsal, ou un gestionnaire de système.

Tels que montrés à la Figure 27, les contrats doivent être établis par le gestionnaire de système. Ils doivent également être maintenus, modifiés et résiliés par le gestionnaire de système. Le gestionnaire de système doit interagir avec les appareils affectés dans le réseau pour accomplir chacune de ces opérations.



Légende

Anglais	Français
Device	Appareil
UAP	UAP
Contract request from UAP to system manager via DMAP	Demande de contrat, de l'UAP vers le gestionnaire de système en passant par le DMAP
DMAP	DMAP
System manager	Gestionnaire de système
Contract response from system manager to UAP via DMAP	Réponse de contrat, du gestionnaire de système vers l'UAP en passant par le DMAP

Figure 27 – Interaction UAP-gestionnaire de système au cours de l'établissement d'un contrat

6.3.11.2.2 Directionnalité de contrat

Les contrats doivent être unidirectionnels, c'est-à-dire qu'un contrat particulier est limité à la communication d'une source vers une destination. Pour la communication dans le sens opposé, un contrat séparé doit être établi.

Pour une communication bidirectionnelle entre deux appareils, chaque appareil doit obtenir du gestionnaire de système un contrat indépendant afin d'envoyer ses messages vers l'autre appareil. Les processus d'application homologues dans ces appareils sont censés être configurés de telle manière qu'ils établissent ces contrats avant de commencer la messagerie dans un sens ou dans l'autre. De telles configurations sont faites soit par le gestionnaire de système si les processus d'application sont les DMAP, soit par des applications d'hôte sur des réseaux d'installation, soit par des outils de maintenance tenus à la main si les processus d'application sont des UAP.

6.3.11.2.3 Définition d'un identificateur de contrat

Un ID de contrat (identificateur de contrat) est un identificateur assigné par le gestionnaire de système qui doit être fourni à la source après que les ressources réseau nécessaires ont été allouées pour assurer la prise en charge de la communication demandée.

L'ID de contrat est pertinent au niveau de la source, car il est utilisé par le gestionnaire de système pour informer chaque couche de protocoles dans la source de la façon de traiter les unités de données de service. Les couches ont besoin de cette information avant d'émettre des SDU vers la destination en passant par le réseau. Le processus d'application demandant un contrat doit récupérer l'ID de contrat assigné et doit l'utiliser pour envoyer des unités de données de protocole en descendant la suite de protocoles. Les configurations des suites de protocoles à chaque couche pour traiter de telles PDU de couche supérieure doivent être référencées à l'ID de contrat. Plus de détails sont fournis en 6.3.11.2.9.

Les ID de contrat sont uniques seulement en ce qui concerne la source, à savoir que le gestionnaire de système peut assigner la même valeur numérique d'ID de contrat à deux appareils indépendants pour prendre en charge leurs demandes indépendantes de contrat. La combinaison d'une adresse IPv6Address source et de son ID de contrat doit être unique à travers le réseau.

L'ID de contrat est également pertinent au niveau du routeur dorsal qui prend en charge la communication prévue pour une destination dans le sous-réseau D pris en charge par le routeur dorsal en question. La raison en est que la DL dans le routeur dorsal a besoin de déterminer comment envoyer la NPDU vers sa destination, en passant par le réseau D. La configuration de la DL dans le routeur dorsal pour traiter de telles NPDU doit être référencée à la combinaison de l'adresse IPv6Address source et de son ID de contrat. Plus de détails sont fournis en 6.3.11.2.9.2.

L'ID de contrat n'est pertinent au niveau d'aucun autre appareil intermédiaire sur le chemin entre la source et la destination.

Alors que les ID de contrat sont des valeurs à 2 octets, le gestionnaire de système doit limiter l'assignation des ID de contrat qui s'inscrivent dans la plage 1..255 aux contrats impliquant un ou plusieurs appareils de terrain, car de tels appareils ont habituellement de plus strictes contraintes de mémoire que les autres appareils, permettant de ce fait à de tels appareils de terrain de stocker des ID de contrat en utilisant un seul octet. L'ID de contrat 0 est réservé pour signifier "noContract" (absence de contrat).

6.3.11.2.4 Architecture prenant en charge la messagerie relative à un contrat

6.3.11.2.4.1 Généralités

Le DMO dans chaque appareil et le SCO dans le gestionnaire de système doivent travailler ensemble pour fournir à chaque appareil des services relatifs à un contrat tels que l'établissement de contrat, la maintenance et la modification de contrat, et la résiliation de contrat.

6.3.11.2.4.2 Traitement de services relatifs à un contrat au sein d'un appareil

Le DMO dans chaque appareil doit être responsable de demander, de maintenir, de modifier et de résilier chaque contrat assigné à l'appareil en question.

N'importe quel processus d'application qui requiert un contrat a besoin d'envoyer la demande au DMO qui à son tour doit envoyer la demande au gestionnaire de système. Après que le contrat a été établi, ce processus d'application ne doit pas essayer d'utiliser des ressources réseau excédant celles qui ont été allouées pour ce contrat. Après que le contrat a été établi, ce processus d'application peut ultérieurement demander une modification ou une résiliation de ce contrat, selon le cas.

L'attribut structuré Contract_Table du DMO peut être accessible directement ou indirectement par le SCO présent dans le gestionnaire de système.

Le gestionnaire de système accède indirectement à l'attribut de structure Contract_Table lorsqu'il envoie à l'appareil une quelconque réponse relative au contrat. L'appareil doit mettre à jour l'attribut structuré Contract_Table du DMO chaque fois qu'une réponse de contrat est reçue en provenance du gestionnaire de système. Une nouvelle entrée dans l'attribut structurée Contracts_Table du DMO doit être créée chaque fois qu'une réponse de contrat associée à la création réussie d'un nouveau contrat est reçue en provenance du gestionnaire de système.

Le gestionnaire de système peut directement lire ou écrire n'importe quel élément présent dans l'attribut structuré Contract_Table du DMO une fois que l'entrée de contrat existe dans l'appareil.

6.3.11.2.4.3 Traitement de services relatifs à un contrat dans le réseau

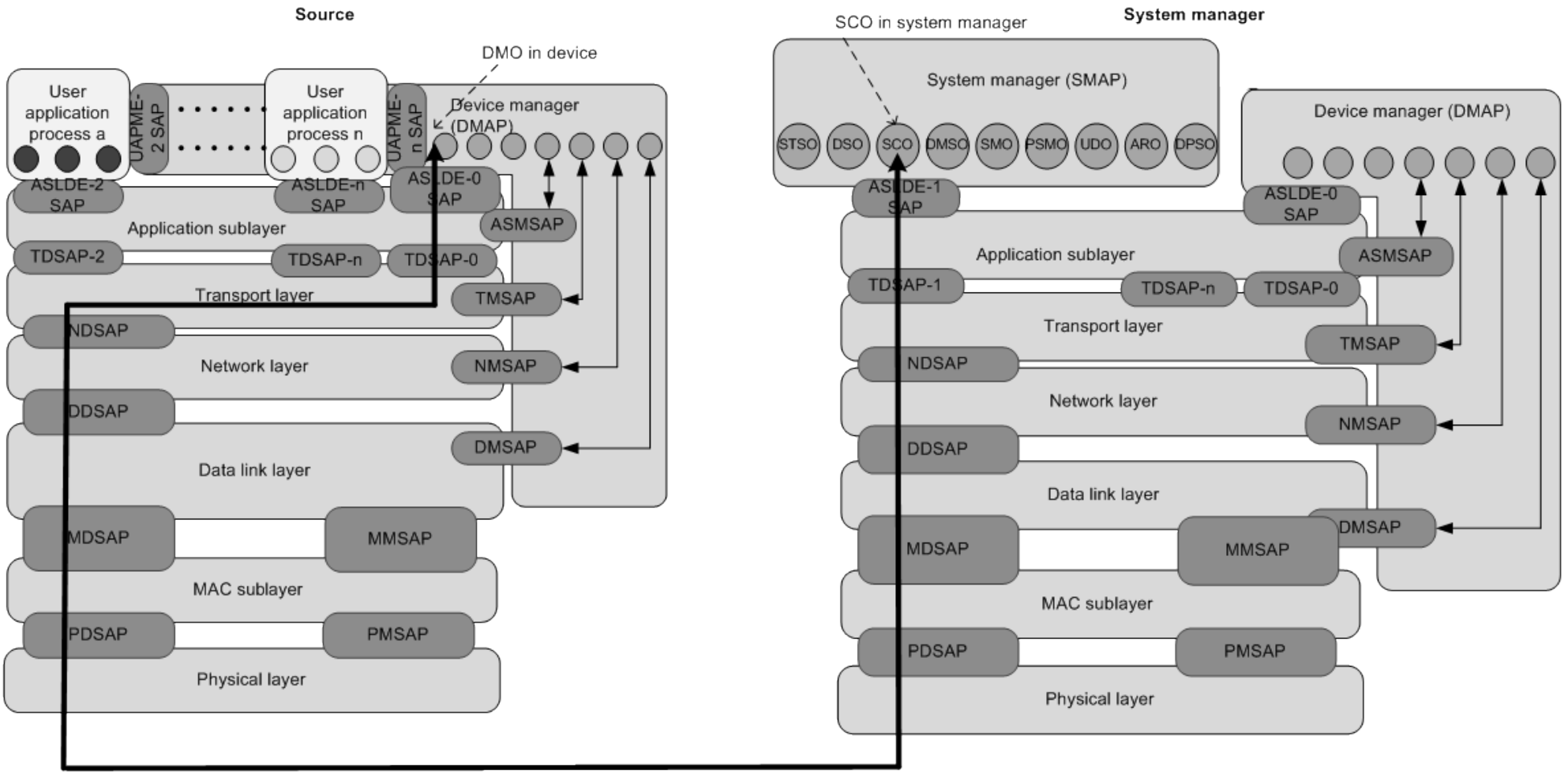
Le SCO dans le gestionnaire de système est responsable d'établir, de maintenir, de modifier et de résilier tous les contrats dans le réseau. Le SCO doit coordonner avec le DMO de chaque appareil l'accomplissement de ces opérations.

6.3.11.2.4.4 System communication configuration object (objet de configuration de communications système)

Les attributs du SCO sont donnés dans le Tableau 26. Les méthodes du SCO sont données dans le Tableau 27.

Tableau 26 – Attributs du SCO dans le gestionnaire de système

Nom du type d'objet normalisé: System communication configuration object (SCO, objet de configuration de communications système)				
Identificateur du type d'objet normalisé: 102				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Réservé pour les éditions futures de la présente norme	1..63	-	-	-



Légende

Anglais	Français
Source	Source
DMO in device	DMO dans l'appareil
User application process a	Processus d'application utilisateur a

Anglais	Français
User application process n	Processus d'application utilisateur n
Application sub-layer	Sous-couche d'application
Transport layer	Couche transport
Network layer	Couche réseau
Data link layer	Couche liaison de données
MAC sub-layer	Sous-couche MAC
Physical layer	Couche physique
System manager	Gestionnaire de système
SCO in system manager	SCO dans le gestionnaire de système
Device manager (DMAP)	Gestionnaire d'appareil (DMAP)
System manager (SMAP)	Gestionnaire de système (SMAP)

Figure 28 – Interaction, relative à un contrat, entre DMO et SCO

6.3.11.2.4.5 Messages relatifs à un contrat

Tous les messages relatifs à un contrat échangés entre le SCO dans le gestionnaire de système et le DMO de l'appareil doivent être des messages C/S de niveau application (à savoir écritures et lectures sur des attributs d'objets normalisés et exécutions sur des méthodes d'objets normalisées). Cela est illustré à la Figure 28.

Les messages relatifs à un contrat incluent des demandes de contrat et des réponses de contrat; ceux-ci sont décrits en 6.3.11.2.5.4.

6.3.11.2.5 Etablissement de contrat

6.3.11.2.5.1 Généralités

Des contrats doivent être établis par le gestionnaire de système lorsqu'il reçoit une demande de contrat. Un processus d'application, qui a besoin de communiquer avec un processus homologue à travers le réseau, émet une demande de contrat à destination du DMAP au sein de l'appareil. Le DMO dans cet appareil demandeur doit envoyer cette demande de contrat au SCO dans le gestionnaire de système. Chaque demande de contrat doit inclure des arguments qui sont utilisés par le SCO pour déterminer l'allocation de ressources réseau nécessaires pour prendre en charge cette demande. Ces arguments sont débattus en 6.3.11.2.5.4.

Si un appareil reçoit une demande de service, mais n'a pas déjà un contrat nécessaire afin d'envoyer la réponse de service, il convient qu'il demande un contrat. L'appareil ne doit pas envoyer la réponse de service tant que la réponse de contrat n'a pas été reçue en provenance du SCO du gestionnaire de système et toutes les ressources nécessaires pour prendre en charge le contrat n'ont pas été configurées avec succès.

Les algorithmes utilisés par le SCO pour déterminer l'allocation nécessaire des ressources réseau ne sont pas spécifiés dans la norme, car ils sont tous internes au gestionnaire de système. Les fournisseurs sont censés mettre en œuvre des algorithmes dans le gestionnaire de système qui peuvent déterminer l'allocation nécessaire de ressources réseau pour les demandes de contrat envoyées par les appareils gérés par le gestionnaire de système en question.

En se basant sur cette détermination, le SCO doit allouer les ressources réseau en communiquant avec les appareils nécessaires dans le réseau et en fournissant à chacun de ceux-ci les configurations nécessaires de suites de protocoles. Cela doit inclure la configuration de la destination et de la source. Des détails sont fournis en 6.3.11.2.6. Des contrats doivent être établis pour la communication tant programmée que non programmée entre les applications.

Comme partie intégrante de la configuration de la source, le SCO doit également fournir l'ID de contrat. Lorsque la source reçoit cette configuration, elle doit utiliser ces informations de contrat pour commencer à émettre les TSDU qui avaient besoin de cette prise en charge des communications. Après que le contrat a été établi, la source ne doit pas essayer d'utiliser des ressources réseau excédant celles qui ont été allouées pour ce contrat.

6.3.11.2.5.2 Relation entre contrats et sessions

Des sessions doivent être établies entre l'accès T dans la source et l'accès T correspondant dans la destination. Toute la communication entre ces accès doit être sécurisée en utilisant une clé T qui est émise par le gestionnaire de sécurité. L'établissement de sessions est décrit en 7.5. Les contrats doivent soutenir la communication entre les processus d'application homologues qui résident sur ces accès T dans la pile de protocoles.

Plusieurs contrats peuvent être établis entre ces processus d'application homologues pour prendre en charge les différents besoins de communication. Chaque processus d'application

dans un appareil étant associé à un accès T, tous ces contrats de ces processus d'application homologues doivent utiliser les mêmes accès T dans la source et dans la destination et, donc, la même clé T doit être utilisée pour sécuriser toute la communication qui se produit en utilisant ces multiples contrats.

La clé T entre les accès T correspondants dans la source et dans la destination a besoin d'être établie avant qu'un contrat ne puisse être utilisé pour envoyer des messages par ces accès T. Donc, avant que le DMO n'envoie la demande de contrat au SCO, il est censé vérifier avec le DSMO dans le DMAP pour voir si une clé T existe entre les accès T correspondants dans la source et la destination. Si une telle clé T n'existe pas, le DSMO est censé envoyer une demande de clé T au gestionnaire de sécurité et obtenir une nouvelle clé T. Cette demande de clé T est décrite en 7.6.3. Si une clé T existe déjà entre les accès T correspondants, le DMO peut immédiatement envoyer la demande de contrat au SCO.

Si, pour une raison quelconque, le gestionnaire de sécurité ou l'appareil lui-même met fin à une clé T existante d'un accès T particulier dans l'appareil, tout contrat qui utilise cet accès T particulier échouera, car le message ne sera pas envoyé en descendant la pile de protocoles dans l'appareil. Dans ce cas, la TL est censée renvoyer une indication d'échec au processus d'application qui a généré le message. Ce processus d'application à son tour est censé envoyer une demande de contrat au DMO qui vérifie avec le DSMO l'existence d'une clé T. Le DSMO peut envoyer une nouvelle demande de clé T au gestionnaire de sécurité, s'il y a lieu.

6.3.11.2.5.3 Appareils qui peuvent demander des contrats

Seule une source de trafic doit présenter une demande de contrat au gestionnaire de système. Les autres appareils dans le réseau ne sont pas autorisés à présenter une demande de contrat pour le compte de la source.

6.3.11.2.5.4 Arguments de demande et de réponse de contrat

Les arguments de demande de contrat sont des éléments d'information qui sont fournis par l'application demandeuse de contrat. Ils sont basés sur le besoin de communication auquel le processus d'application demandeur est intéressé.

Il convient de concevoir les applications de telle sorte qu'elles demandent seulement la plus petite quantité de prise en charge de communication nécessaire pour satisfaire à leurs besoins de communication. Par exemple, il convient qu'une application qui a besoin d'éditer des messages périodiques toutes les minutes ne demande pas une période de 10 s.

Les arguments de réponse de contrat doivent inclure l'ID de contrat et autres informations. Ces informations sont destinées à fournir la configuration de suite de protocoles de base nécessaire au niveau de la source.

Les arguments inclus dans les messages de demande et de réponse de contrat sont décrits dans le Tableau 27.

Tableau 27 – Méthode du SCO pour l'établissement, la modification ou le renouvellement de contrat *(1 de 9)*

Nom du type d'objet normalisé: System communication configuration (SCO, objet de configuration de communications système)				
Identificateur du type d'objet normalisé: 102				
Nom de la méthode	**ID de la méthode**	**Description de la méthode**		
Contract_ Establishment _ Modification_ Renewal	1	Méthode permettant d'établir un nouveau contrat, de modifier ou de renouveler un contrat existant en envoyant une demande au gestionnaire de système		
	Arguments d'entrée			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	Contract_Request_ID	Unsigned8	Valeur numérique, assignée de manière unique par l'appareil envoyant la demande au gestionnaire de système, pour identifier la demande faite. Prend la valeur par défaut zéro et se réinitialise à zéro. S'incrémente avec chaque utilisation. Cet ID doit être répété si exactement la même demande est renvoyée de nouveau en raison de l'absence de réponse. Repasse par zéro
	2	Request_Type	Unsigned8	Type de demande de contrat envoyée au gestionnaire de système. Valeurs nommées: 0: nouveau contrat; 1: modification de contrat; 2: renouvellement de contrat. Certains des arguments d'entrée ci-dessous ne sont pas applicables sur la base de cette valeur d'argument; voir le Tableau 28 pour les détails
	3	Contract_ID	Unsigned16	ID de contrat existant qui a besoin d'être modifié ou renouvelé
	4	Communication_ Service_Type	Unsigned8	Type de service de communication pour lequel le contrat est demandé. Valeurs nommées: 0: périodique/programmé; 1: non périodique/non programmé. Certains des arguments d'entrée ci-dessous ne sont pas applicables sur la base de cette valeur d'argument; voir le Tableau 28 pour les détails
	5	Source_SAP	Unsigned16	TDSAP dans la source qui utilisera ce contrat, une fois qu'il est assigné, pour envoyer des messages d'application en descendant la pile de protocoles
	6	Destination_Address	IPv6Address	Adresse de l'appareil vers laquelle la source souhaite envoyer des messages d'application; noter que ces informations peuvent être fournies à la source au cours de la configuration ou au cours de la configuration du processus d'application
	7	Destination_SAP	Unsigned16	TDSAP dans la destination qui sera utilisée pour envoyer ces messages vers l'AL; noter que ces informations peuvent être fournies à la source au cours de la configuration ou au cours de la configuration du processus d'application

Tableau 27 *(2 de 9)*

Nom du type d'objet normalisé: System communication configuration (SCO, objet de configuration de communications système)				
Identificateur du type d'objet normalisé: 102				
	Arguments d'entrée			
	Numéro de l'argument	Nom de l'argument	Type de l'argument (type et taille des données)	Description de l'argument
	8	Contract_Negotiability	BitString8	Détermine si le gestionnaire de système peut changer le contrat demandé pour satisfaire aux ressources de réseau disponibles et si le gestionnaire de système peut révoquer ce contrat pour rendre des ressources disponibles pour des contrats de plus haute priorité. Indices nommés: 0: non révocable; 1: non négociable; 2..7: réservés. La négociabilité de contrat est décrite en 6.3.11.2.7.2
	9	Contract_Expiration_Time	Unsigned32	Détermine la durée pendant laquelle il convient que le gestionnaire de système conserve le contrat avant qu'il ne soit résilié; unités en secondes
	10	Contract_Priority	Unsigned8	Demande une priorité de base pour tous les messages envoyés en utilisant le contrat. Valeurs nommées: 0: meilleur effort mis en file d'attente; 1: temps réel séquentiel; 2: tampon temps réel; 3: commande réseau. La priorité de contrat est décrite en 6.3.11.2.7.3
	11	Payload_Size	Unsigned16	Indique la taille maximale en octets (représentée comme taille d'APDU) de charge utile que la source est susceptible de vouloir transmettre. Plage de valeurs: 3..1 252

Tableau 27 *(3 de 9)*

Nom du type d'objet normalisé: System communication configuration (SCO, objet de configuration de communications système)				
Identificateur du type d'objet normalisé: 102				
	Arguments d'entrée			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	12	Reliability_And_Publish DoNotAutoRetransmit	Unsigned8	PublishDoNotAutoRetransmit: Le bit 0 indique si la retransmission d'anciennes données d'édition n'est pas prise en charge du fait que la valeur en tampon précédente est toujours écrasée en écriture par les nouvelles données d'édition. Le bit 0 n'est applicable qu'aux communications périodiques et est égal à 0 pour les communications non périodiques (voir 12.12.2 pour une description). [a] Fiabilité: Les bits 1..7 indiquent la fiabilité demandée pour livrer à la destination les APDU émises. Bit 0: Unsigned1: Valeurs nommées: 0: retransmission auto; 1: aucune retransmission auto. Bits 1..7: Unsigned7: Valeurs nommées: 0: faible; 1: moyenne; 2: élevée
	13	Requested_Period	Integer16	Utilisé pour la communication périodique pour identifier la période d'édition souhaitée dans la demande de contrat. Plage valide: Une valeur de $N > 0$ spécifie une période de N s tandis qu'une valeur de $N < 0$ spécifie une période de -1/N s. $N = 0$ désactive la communication
	14	Requested_Phase	Unsigned8	Utilisé pour la communication périodique pour identifier la phase souhaitée (dans les limites de la période d'édition) des publications dans la demande de contrat. Plage valide: 0..99; toute autre valeur indique que l'appareil se préoccupe seulement de la période et ne se préoccupe pas de la phase. Voir 6.3.11.2.7.4

Tableau 27 *(4 de 9)*

Nom du type d'objet normalisé: System communication configuration (SCO, objet de configuration de communications système)				
Identificateur du type d'objet normalisé: 102				
	Arguments d'entrée			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	15	Requested_Deadline	Unsigned16	Utilisé pour la communication périodique pour identifier le retard maximal de transport de bout en bout souhaité. Unité:10 ms
	16	Committed_Burst	Integer16	Utilisé pour la communication non périodique pour identifier le taux à long terme qui a besoin d'être pris en charge pour les messages client/serveur ou source/puits. Plage valide: Une valeur de $N > 0$ spécifie une fréquence d'APDU par seconde moyenne tandis qu'une valeur de $N < 0$ spécifie une fréquence de -1/N APDU par seconde. $N = 0$ n'est pas valide
	17	Excess_Burst	Integer16	Utilisé pour la communication non périodique pour identifier le taux à court terme qui a besoin d'être pris en charge pour les messages client/serveur ou source/puits. Plage valide: voir argument d'entrée 16
	18	Max_Send_Window_ Size	Unsigned8	Utilisé pour la communication non périodique pour identifier le nombre maximal de demandes client pouvant être simultanément en attente d'une réponse
	Arguments de sortie			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	Contract_Request_ID	Unsigned8	L'argument d'entrée Contract_Request_ID qui a été reçu dans la demande de contrat correspondante est utilisé comme cet argument de sortie

Tableau 27 *(5 de 9)*

Nom du type d'objet normalisé: System communication configuration (SCO, objet de configuration de communications système)				
Identificateur du type d'objet normalisé: 102				
	Arguments de sortie			
	Numéro de l'argument	Nom de l'argument	Type de l'argument (type et taille des données)	Description de l'argument
	2	Response_Code	Unsigned8	Indique si le gestionnaire de système a réussi à prendre en charge la demande de contrat; indique si la source peut utiliser le contrat immédiatement ou si elle doit attendre; indique également si la communication demandée est prise en charge telle quelle ou si le gestionnaire de système a négocié la demande à la baisse. Valeurs nommées: 0: succès avec effet immédiat; 1: succès avec effet différé; 2: succès avec effet immédiat, mais négocié à la baisse; 3: succès avec effet différé, mais négocié à la baisse; 4: échec sans autre conseil; 5: échec avec conseil de répétition de tentative; 6: échec avec conseil de répétition de tentative et de négociation. Selon la valeur de cet argument, certains des arguments de sortie ci-dessous ne sont pas applicables. Voir le Tableau 29 pour les détails et le 6.3.11.2.12 pour les scénarios d'échec
	3	Contract_ID	Unsigned16	Valeur numérique assignée de façon unique par le gestionnaire de système au contrat étant établi et envoyé à la source. Les ID de contrat sont uniques par appareil. Selon les ressources demandées, plusieurs ID de demande de contrat issus d'un appareil peuvent faire l'objet d'un mapping à un seul ID de contrat. Dans l'appareil, l'ID de contrat est passé dans le champ commande de DSAP de chaque couche et est utilisé pour consulter les actions contractées qui doivent être prises sur la PDU associée pendant qu'elle descend dans la suite de protocoles à chaque couche (valeur 0 réservée signifiant l'absence de contrat)
	4	Communication_Service_Type	Unsigned8	Type de service de communication pris en charger par ce contrat. Unsigned8: voir argument d'entrée 4. Certains des arguments de sortie ci-dessous ne sont pas applicables sur la base de cette valeur d'argument; voir le Tableau 29 pour les détails

Tableau 27 *(6 de 9)*

Nom du type d'objet normalisé: System communication configuration (SCO, objet de configuration de communications système)				
Identificateur du type d'objet normalisé: 102				
	Arguments de sortie			
	Numéro de l'argument	Nom de l'argument	Type de l'argument (type et taille des données)	Description de l'argument
	5	Contract_Activation_Time	TAINetworkTime	Heure de début pour que la source commence à utiliser le contrat assigné
	6	Assigned_Contract_Expiration_Time	Unsigned32	Détermine la durée pendant laquelle le gestionnaire de système doit conserver le contrat avant qu'il ne soit résilié. Unité: s
	7	Assigned_Contract_Priority	Unsigned8	Etablit une priorité de base pour tous les messages envoyés en utilisant le contrat. Voir argument d'entrée 10. La priorité de contrat est décrite en 6.3.11.2.7.3.
	8	Assigned_Max_TSDU_Size	Unsigned16	Indique la TSDU maximale prise en charge en octets qui peut être convertie par la source en taille d'APDU maximale en prenant en compte les tailles de la TL, de la sécurité, des en-têtes d'AL et du TMIC. Plage valide: 70..1 280. Le gestionnaire de système doit prendre en compte l'attribut constant Max_NSDU_Size rapporté par les NLMO de la source et de la destination (voir Tableau 206) lors de la détermination de la valeur de cet argument. La fragmentation est réalisé à la NL si la NPDU excède la taille maximale d'une DSDU. La fragmentation et le réassemblage sont décrits en 10.2.5
	9	Assigned_Reliability_And_PublishDoNotAutoRetransmit	Unsigned8	Voir argument d'entrée 12
	10	Assigned_Period	Integer16	voir argument d'entrée 13
	11	Assigned_Phase	Unsigned8 Plage valide: 0..99	Utilisé pour la communication périodique pour identifier la phase assignée (dans les limites de la période d'édition) des publications dans le contrat

Tableau 27 *(7 de 9)*

Nom du type d'objet normalisé: System communication configuration (SCO, objet de configuration de communications système)				
Identificateur du type d'objet normalisé: 102				
	Arguments de sortie			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	12	Assigned_Deadline	Unsigned16	Utilisé pour la communication périodique pour identifier le retard maximal de transport de bout en bout pris en charge par le contrat assigné. Unit: 10 ms
	13	Assigned_Committed_ Burst	Integer16	Utilisé pour la communication non périodique pour identifier le taux à long terme qui est pris en charge pour les messages client/serveur ou source/puits. Plage valide: Une valeur de $N > 0$ spécifie une fréquence d'APDU par seconde moyenne tandis qu'une valeur de $N < 0$ spécifie une fréquence de -1/N APDU par seconde. $N = 0$ n'est pas valide
	14	Assigned_Excess_Burst	Integer16	Utilisé pour la communication non périodique pour identifier le taux à court terme qui est pris en charge pour les messages client/serveur ou source/puits. Plage valide: Une valeur de $N > 0$ spécifie une fréquence d'APDU par seconde moyenne tandis qu'une valeur de $N < 0$ spécifie une fréquence de -1/N APDU par seconde. $N = 0$ n'est pas valide
	15	Assigned_Max_Send_ Window_Size	Unsigned8	Utilisé pour la communication non périodique pour identifier le nombre maximal permis de demandes client pouvant être simultanément en attente d'une réponse
	16	Retry_Backoff_Time	Unsigned16	Utilisé dans le cas du code de réponse = échec avec conseil de répétition de tentative ou échec avec conseil de répétition de tentative et de négociation; indique la quantité de temps dont il convient que la source se replie avant d'envoyer de nouveau la demande de contrat; unités en secondes; les scénarios d'échec sont décrits en 6.3.11.2.12

Tableau 27 *(8 de 9)*

Nom du type d'objet normalisé: System communication configuration (SCO, objet de configuration de communications système)				
Identificateur du type d'objet normalisé: 102				
	Arguments de sortie			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	17	Negotiation_Guidance	BitString8	Utilisé dans le cas d'un code de réponse = échec avec conseil de répétition de tentative et de négociation; indique la valeur de Contract_Negotiability pouvant être prise en charge par le gestionnaire de système. Attributions d'indices: voir argument d'entrée 8 Les scénarios d'échec sont décrits en 6.3.11.2.12
	18	Supportable_Contract_ Priority	Unsigned8	Indique la priorité de base pouvant être prise en charge par le gestionnaire de système pour tous les messages envoyés en utilisant le contrat. Unsigned8: voir argument d'entrée 10
	19	Supportable_max_ TSDU_Size	Unsigned16 Plage valide: 70..1 280	Indique la NSDU maximale pouvant être prise en charge par le gestionnaire de système; unités en octets
	20	Supportable_Reliability_ And_PublishDoNot AutoRetransmit	Unsigned8	Voir argument d'entrée 12
	21	Supportable_Period	Integer16	Utilisé pour la communication périodique pour identifier la période d'édition pouvant être prise en charge par le gestionnaire de système. Plage valide: voir argument d'entrée 13
	22	Supportable_Phase	Unsigned8 Plage valide: 0..99	Utilisé pour la communication périodique pour identifier la phase (dans les limites de la période d'édition) des publications pouvant être prises en charge par le gestionnaire de système
	23	Supportable_Deadline	Unsigned16	Utilisé pour la communication périodique pour identifier le retard maximal de transport de bout en bout pouvant être pris en charge par le gestionnaire de système. Unité:10 ms

Tableau 27 *(9 de 9)*

Nom du type d'objet normalisé: System communication configuration (SCO, objet de configuration de communications système)				
Identificateur du type d'objet normalisé: 102				
	Arguments de sortie			
	Numéro de l'argument	Nom de l'argument	Type de l'argument (type et taille des données)	Description de l'argument
	24	Supportable_Committed_Burst	Integer16	Utilisé pour la communication non périodique pour identifier le taux à long terme pouvant être pris en charge pour les messages client/serveur ou source/puits. Plage valide: voir argument d'entrée 16
	25	Supportable_Excess_Burst	Integer16	Utilisé pour la communication non périodique pour identifier le taux à court terme pouvant être pris en charge pour les messages client/serveur ou source/puits. Plage valide: Une valeur de $N > 0$ spécifie une fréquence d'APDU par seconde moyenne tandis qu'une valeur de $N < 0$ spécifie une fréquence de -1/N APDU par seconde. $N = 0$ n'est pas valide
	26	Supportable_Max_Send_Window_Size	Unsigned8	Utilisé pour la communication non périodique pour identifier le nombre maximal pouvant être pris en charge de demandes client pouvant être simultanément en attente d'une réponse
[a] Le codage de cet attribut est l'inverse de l'attribut associé 7 du Tableau 265.				

Le Tableau 27 contient également des arguments d'entrée et de sortie qui doivent être utilisés pour la modification de contrat et le renouvellement de contrat. La modification de contrat et le renouvellement de contrat sont débattus en 6.3.11.2.11.

Le Tableau 27 contient également des arguments de sortie qui doivent être utilisés pour des scénarios d'échec lorsque le gestionnaire de système n'est pas capable de prendre en charge la demande de contrat. Ces scénarios d'échec sont débattus en 6.3.11.2.12.

Certains des arguments d'entrée dans le Tableau 27 ne sont pas applicables lorsqu'il est donné certaines valeurs aux arguments Request_Type et/ou Communication_Service_Type et ne doivent donc pas être inclus dans la demande. Ces informations sont fournies dans le Tableau 28.

Tableau 28 – Utilisation des arguments d'entrée pour la méthode du SCO pour l'établissement, la modification ou le renouvellement de contrat

Argument d'entrée	Non applicable à	
	Valeur Request_Type	Valeur Communication_Service_Type
Contract_Request_ID	-	-
Request_Type	-	-
Contract_ID	0	-
Communication_Service_Type	-	-
Source_SAP	-	-
Destination_Address	-	-
Destination_SAP	-	-
Contract_Negotiability	-	-
Contract_Expiration_Time	-	-
Contract_Priority	-	-
Payload_Size	-	-
Reliability_And_PublishDoNotAutoRetransmit	-	-
Requested_Period	-	1
Requested_Phase	-	1
Requested_Deadline	-	1
Committed_Burst	-	0
Excess_Burst	-	0
Max_Send_Window_Size	-	0

Certains des arguments de sortie dans le Tableau 27 ne sont pas applicables lorsqu'il est donné certaines valeurs aux arguments Response_Code et/ou Communication_Service_Type et ne doivent donc pas être inclus dans la réponse. Ces informations sont fournies dans le Tableau 29.

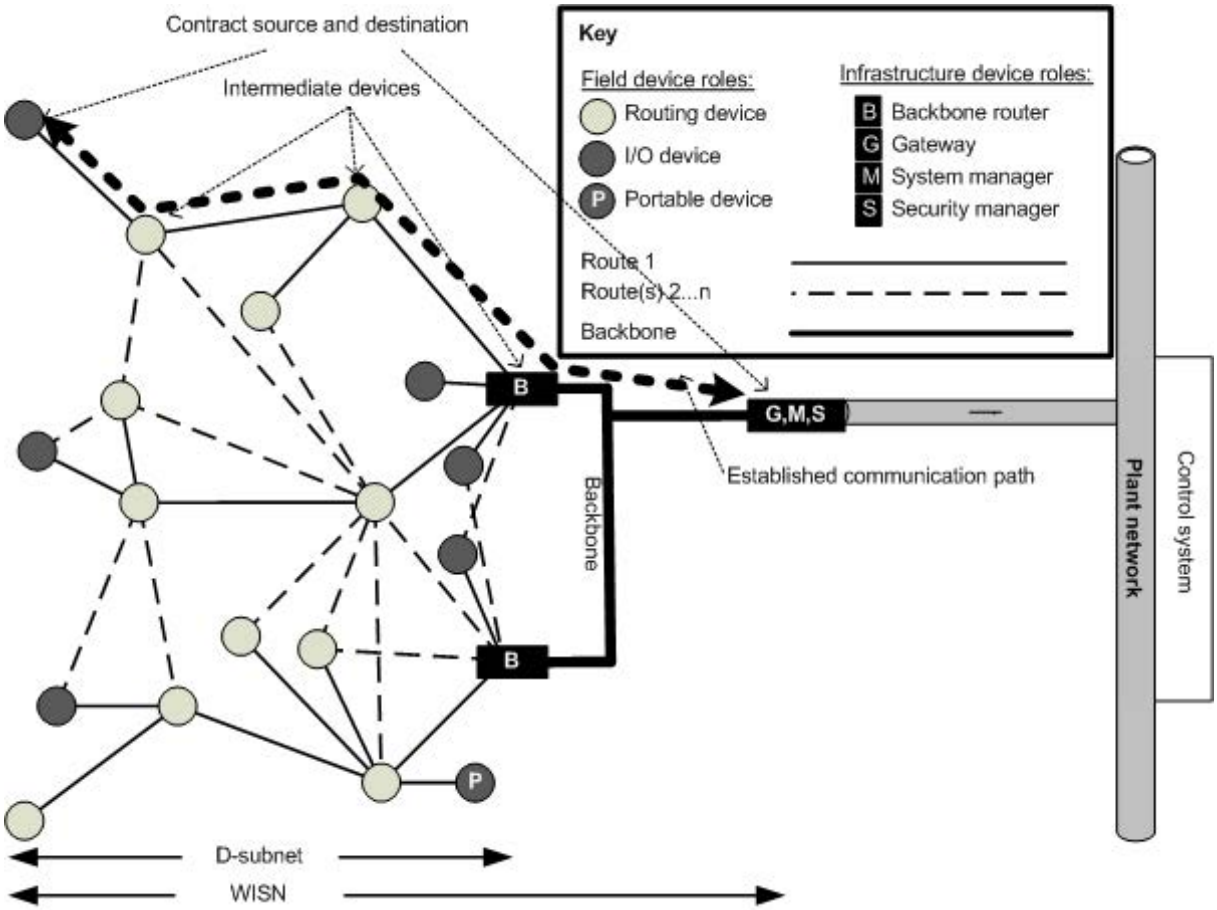
Tableau 29 – Utilisation des arguments de sortie pour la méthode du SCO pour l'établissement, la modification ou le renouvellement de contrat

Argument de sortie	Non applicable à	
	Valeur Response_Code	Valeur Communication_Service_Type
Contract_Request_ID	-	-
RESPONSE_CODE	-	-
Contract_ID	4, 5, 6	-
Communication_Service_Type	4, 5	-
Contract_Activation_Time	0, 2, 4, 5, 6	-
Assigned_Contract_Expiration_Time	4, 5, 6	-
Assigned_Contract_Priority	4, 5, 6	-
Assigned_Max_TSDU_Size	4, 5, 6	-
Assigned_Reliability_And_PublishDoNotAutoRetransmit	4, 5, 6	-
Assigned_Period	4, 5, 6	1
Assigned_Phase	4, 5, 6	1
Assigned_Deadline	4, 5, 6	1
Assigned_Committed_Burst	4, 5, 6	0
Assigned_Excess_Burst	4, 5, 6	0
Assigned_Max_Send_Window_Size	4, 5, 6	0
Retry_Backoff_Time	0, 1, 2, 3, 4	-
Negotiation_Guidance	0, 1, 2, 3, 4, 5	-
Supportable_Contract_Priority	0, 1, 2, 3, 4, 5	-
Supportable_max_TSDU_Size	0, 1, 2, 3, 4, 5	-
Supportable_Reliability_And_PublishDoNotAutoRetransmit	0, 1, 2, 3, 4, 5	-
Supportable_Period	0, 1, 2, 3, 4, 5	1
Supportable_Phase	0, 1, 2, 3, 4, 5	1
Supportable_Deadline	0, 1, 2, 3, 4, 5	1
Supportable_Committed_Burst	0, 1, 2, 3, 4, 5	0
Supportable_Excess_Burst	0, 1, 2, 3, 4, 5	0
Supportable_Max_Send_Window_Size	0, 1, 2, 3, 4, 5	0

6.3.11.2.6 Configuration de suites de protocoles

6.3.11.2.6.1 Généralités

En tant que partie intégrante de l'établissement de contrat, le SCO doit configurer les appareils nécessaires dans le réseau en fournissant à chacun de ceux-ci les configurations de suites de protocoles nécessaires. Cela doit inclure la configuration de la destination et de la source, telle que montrée à la Figure 29.



Légende

Anglais	Français
Key	Légende
Field device roles	Rôles d'appareil de terrain
Routing device	Appareil de routage
I/O device	Appareil E/S
Portable device	Appareil portatif
Infrastructure device roles	Rôles d'appareil d'infrastructure
B Backbone router	B Routeur dorsal
G Gateway	G Passerelle
M System manager	M Gestionnaire de système
S Security manager	S Gestionnaire de sécurité
Route 1	Chemin 1
Route(s) 2...n	Chemin(s) 2...n
Backbone	Dorsale
Contract source and destination	Source et destination du contrat
Intermediate devices	Appareils intermédiaires
Established communication path	Chemin de communication établi
Backbone	Dorsale
Control system	Système de commande
Plant network	Réseau d'installation
D-subnet	Sous-réseau D
WISN	WISN

Figure 29 – Source, destination et appareils intermédiaires du contrat

Les appareils intermédiaires dans le réseau qui prennent en charge le chemin de communication établi entre la source et la destination doivent être configurés par le SCO. De tels appareils intermédiaires le long du chemin peuvent inclure aussi bien des routeurs de terrain que des routeurs dorsaux.

6.3.11.2.6.2 Configuration des routeurs de terrain intermédiaires

La configuration des routeurs de terrain intermédiaires doit être limitée à la DLE dans chaque routeur de terrain, car le message de la source vers la destination ne traverse que par la DLE de chaque routeur de terrain le long du chemin.

Les attributs et les méthodes définis pour le DLMO des routeurs de terrain doivent être utilisés par le gestionnaire de système pour configurer les routeurs de terrain intermédiaires.

6.3.11.2.6.3 Configuration des routeurs dorsaux intermédiaires

La configuration des routeurs dorsaux intermédiaires doit être limitée à la NL et, dans certains cas, à la DLE dans chaque routeur dorsal, car le message de la source vers la destination traverse par la DLE dans le cas des routeurs dorsaux qui appartiennent aux sous-réseaux D de source et de destination correspondants et par la NLE de chaque routeur dorsal le long du chemin.

Les attributs et les méthodes définis pour le DLMO et le NLMO des routeurs dorsaux doivent être utilisés par le gestionnaire de système pour configurer les routeurs dorsaux intermédiaires.

6.3.11.2.6.4 Configuration de la destination

La configuration de la destination doit inclure la configuration de toutes les couches de protocoles. Les attributs et les méthodes définis pour le DLMO, le NLMO et le TLMO doivent être utilisés par le gestionnaire de système pour configurer la destination.

6.3.11.2.6.5 Configuration de la source

Les arguments de sortie décrits dans le Tableau 27 sont utilisés à diverses couches de la source pour déterminer le traitement des PDU appartenant à ce contrat.

Les attributs et les méthodes définis pour le DLMO, le NLMO et le TLMO doivent être utilisés par le gestionnaire de système pour configurer la source.

Une réponse de contrat doit être envoyée à la source soit après que toutes les ressources réseau nécessaires ont été configurées, soit après que le gestionnaire de système a déterminé le temps qu'il faudrait pour configurer toutes les ressources réseau. Selon la situation, le gestionnaire de système doit indiquer si le contrat assigné peut être utilisé avec effet immédiat ou avec effet différé. Le diagramme de séquence de message à la Figure 30 illustre le cas de l'effet immédiat.

Après que le contrat a été établi, la source ne doit pas essayer d'utiliser des ressources réseau excédant celles qui ont été allouées pour ce contrat.

6.3.11.2.6.6 Informations de contrat dans l'objet de gestion d'appareil

Le DMO dans la source doit maintenir une liste de tous les contrats assignés, et ce, en utilisant l'attribut de Contracts_Table. Cet attribut doit être basé sur la structure de données Contract_Data. Lorsqu'un nouveau contrat est établi, une nouvelle rangée doit être ajoutée à cet attribut Contracts_Table avec les informations de contrat appropriées. Lorsqu'un contrat existant est modifié ou résilié, la rangée correspondante doit être modifiée ou supprimée dans cet attribut Contracts_Table.

Le SCO peut également modifier les paramètres des attributs Contract_Table en y accédant directement sans échanger les structures Contract_Data entières.

Les éléments de la structure de données Contract_Data sont définis dans le Tableau 30.

Tableau 30 – Structure de données Contract_Data *(1 de 3)*

Nom du type de données normalisé: Contract_Data		
Code du type de données normalisé: 401		
Nom de l'élément	**Identificateur de l'élément**	**Type de l'élément**
Contract_ID*	1	Type: Unsigned16 Classification: Static Accessibilité: Read/write Valeurs nommées: 0: aucun contrat Cet élément est le même que l'argument de sortie Contract_ID en Tableau 27. * Cet élément est utilisé comme champ indice pour les méthodes décrites dans le Tableau 33 et dans le Tableau 34
Contract_Status	2	Type: Unsigned8 Classification: Static Accessibilité: Read Only Valeurs nommées: 0: succès avec effet immédiat; 1: succès avec effet différé; 2: succès avec effet immédiat, mais négocié à la baisse; 3: succès avec effet différé, mais négocié à la baisse. Cet élément est relatif à l'argument de sortie Response_Code en Tableau 27
Communication_Service_Type	3	Type: Unsigned8 Classification: Static Accessibilité: Read/write Valeurs nommées: 0: communication périodique/programmée; 1: communication non périodique/non programmée. Cet élément est le même que l'argument de sortie Communication_Service_Type en Tableau 27
Contract_Activation_Time	4	Type: TAINetworkTime Classification: Static Accessibilité: Read/write Cet élément est le même que l'argument de sortie Contract_Activation_Time en Tableau 27
Source_SAP	5	Type: Unsigned16 Classification: Static Accessibilité: Read/write Cet élément est le même que l'argument de sortie Source_SAP en Tableau 27

Tableau 30 *(2 de 3)*

Nom du type de données normalisé: Contract_Data		
Code du type de données normalisé: 401		
Nom de l'élément	**Identificateur de l'élément**	**Type de l'élément**
Destination_Address	6	Type: IPv6Address Classification: Static Accessibilité: Read/write Cet élément est le même que l'argument de sortie Destination_Address en Tableau 27
Destination_SAP	7	Type: Unsigned16 Classification: Static Accessibilité: Read/write Cet élément est le même que l'argument de sortie Destination_SAP en Tableau 27
Assigned_Contract_Expiration_Time	8	Type: Unsigned32 Classification: Static Accessibilité: Read/write Unit: 1 s Cet élément est le même que l'argument de sortie Assigned_Contract_Expiration_Time en Tableau 27
Assigned_Contract_Priority	9	Type: Unsigned8 Classification: Static Accessibilité: Read/write Valeurs nommées: 0: meilleur effort mis en file d'attente; 1: temps réel séquentiel; 2: tampon temps réel; 3: commande réseau. Cet élément est le même que l'argument de sortie Assigned_Contract_Priority en Tableau 27
Assigned_Max_TSDU_Size	10	Type: Unsigned16 Classification: Static Accessibilité: Read/write Plage valide: 70..1 280 Cet élément est le même que l'argument de sortie Assigned_Max_TSDU_Size en Tableau 27
Assigned_Reliability_And_ PublishDoNotAutoRetransmit	11	Type: Unsigned8 Classification: Static Accessibilité: Read/write Plage valide: Bit 0: 0 -- retransmission auto systématique [a] Bits 1..7:: Valeurs nommées: 0: faible; 1: moyenne; 2: élevée. Cet élément est le même que l'argument de sortie Assigned_Reliability_And_PublishDoNotAutoRetransmit en Tableau 27

Tableau 30 *(3 de 3)*

Nom du type de données normalisé: Contract_Data		
Code du type de données normalisé: 401		
Nom de l'élément	**Identificateur de l'élément**	**Type de l'élément**
Assigned_Period	12	Type: Integer16 Classification: Static Accessibilité: Read/write Plage valide: Une valeur de N > 0 spécifie une période de N s tandis qu'une valeur de N < 0 spécifie une période de -1/N s. N = 0 n'est pas valide. Cet élément est le même que l'argument de sortie Assigned_Period en Tableau 27
Assigned_Phase	13	Type: Unsigned8 Classification: Static Accessibilité: Read/write Plage valide: 0..99 Cet élément est le même que l'argument de sortie Assigned_Phase en Tableau 27
Assigned_Deadline	14	Type: Unsigned16 Classification: Static Accessibilité: Read/write Unité:10 ms Cet élément est le même que l'argument de sortie Assigned_Deadline en Tableau 27
Assigned_Committed_Burst	15	Type: Integer16 Classification: Static Accessibilité: Read/write Plage valide: Une valeur de N > 0 spécifie une fréquence d'APDU par seconde moyenne tandis qu'une valeur de N < 0 spécifie une fréquence de -1/N APDU par seconde. N = 0 n'est pas valide. Cet élément est le même que l'argument de sortie Assigned_Committed_Burst en Tableau 27
Assigned_Excess_Burst	16	Type: Integer16 Classification: Static Accessibilité: Read/write Plage valide: Une valeur de N > 0 spécifie une fréquence d'APDU par seconde moyenne tandis qu'une valeur de N < 0 spécifie une fréquence de -1/N APDU par seconde. N = 0 n'est pas valide. Cet élément est le même que l'argument de sortie Assigned_Excess_Burst en Tableau 27
Assigned_Max_Send_Window_Size	17	Type: Unsigned8 Classification: Static Accessibilité: Read/write Cet élément est le même que l'argument de sortie Assigned_Max_Send_Window_Size en Tableau 27
[a] Le codage de cet attribut est l'inverse de l'attribut associé 7 du Tableau 265.		

6.3.11.2.6.7 Configuration de nouvel appareil

Le processus pour un nouvel appareil à rattacher est décrit en 7.4. En tant que partie intégrante du processus de rattachement pour un nouvel appareil, un contrat entre le nouvel appareil et le gestionnaire de système doit être établi.

Le nouvel appareil doit utiliser la méthode Proxy_System_Manager_Contract définie pour le DMO du routeur d'annonce pour envoyer cette demande de contrat, qui est alors transmise au gestionnaire de système, et pour obtenir la réponse de contrat issue du gestionnaire de système par l'intermédiaire du routeur d'annonce. La méthode Proxy_System_Manager_Contract est définie dans le Tableau 20. Le routeur d'annonce doit utiliser la méthode System_Manager_Contract définie dans le DMSO pour transmettre cette demande de contrat et pour recevoir la réponse de contrat associée au processus de rattachement de ce nouvel appareil. La méthode System_Manager_Contract est définie dans le Tableau 24. Le DMSO travaille avec le SCO pour générer cette réponse de contrat. Lorsque le nouvel appareil obtient cette réponse de contrat, une nouvelle rangée doit être ajoutée à l'attribut Contracts_Table dans le DMO du nouvel appareil avec les informations pertinentes de contrat.

Les arguments de sortie dans ces deux méthodes doivent être basés sur la structure de données New_Device_Contract_Response. Les éléments de la structure de données New_Device_Contract_Response sont définis dans le Tableau 31.

Tableau 31 – Structure de données New_Device_Contract_Response *(1 de 2)*

Nom du type de données normalisé: New_Device_Contract_Response		
Code du type de données normalisé: 405		
Nom de l'élément	**Identificateur de l'élément**	**Type de l'élément**
Contract_ID	1	Type: Unsigned16 Classification: Static Accessibilité: Read/write Valeurs nommées: 0: aucun contrat Cet élément est relatif à l'argument de sortie Contract_ID en Tableau 27
Assigned_Max_TSDU_Size	2	Type: Unsigned16 Classification: Static Accessibilité: Read/write Plage valide: 70..1 280 Cet élément est relatif à l'argument de sortie Assigned_Max_TSDU_Size en Tableau 27
Assigned_Committed_Burst	3	Type: Integer16 Classification: Static Accessibilité: Read/write Plage valide: Une valeur de N > 0 spécifie une fréquence d'APDU par seconde moyenne tandis qu'une valeur de N < 0 spécifie une fréquence de -1/N APDU par seconde. N = 0 n'est pas valide. Cet élément est relatif à l'argument de sortie Assigned_Committed_Burst en Tableau 27
Assigned_Excess_Burst	4	Type: Integer16 Classification: Static Accessibilité: Read/write Plage valide: Une valeur de N > 0 spécifie une fréquence d'APDU par seconde moyenne tandis qu'une valeur de N < 0 spécifie une fréquence de -1/N APDU par seconde. N = 0 n'est pas valide. Cet élément est relatif à l'argument de sortie Assigned_Excess_Burst en Tableau 27
Assigned_Max_Send_Window_Size	5	Type: Unsigned8 Classification: Static Accessibilité: Read/write Cet élément est relatif à l'argument de sortie Assigned_Max_Send_Window_Size en Tableau 27
NL_Header_Include_Contract_Flag	6	Type: Boolean1 Classification: Static Accessibilité: Read/write Cet attribut se rapporte à l'élément correspondant en Tableau 208 et est utilisé pour la configuration de la NL du nouvel appareil afin d'utiliser le contrat affecté au nouvel appareil

Tableau 31 *(2 de 2)*

Nom du type de données normalisé: New_Device_Contract_Response		
Code du type de données normalisé: 405		
Nom de l'élément	**Identificateur de l'élément**	**Type de l'élément**
NL_Next_Hop	7	Type: IPv6Address Classification: Static Accessibilité: Read/write Cet attribut se rapporte à l'élément correspondant en Tableau 208 et est utilisé pour la configuration de la NL du nouvel appareil afin d'utiliser le contrat affecté au nouvel appareil
NL_NWK_HopLimit	8	Type: Unsigned8 Classification: Static Accessibilité: Read/write Cet attribut se rapporte à l'élément correspondant en Tableau 208 et est utilisé pour la configuration de la NL du nouvel appareil afin d'utiliser le contrat affecté au nouvel appareil
NL_Outgoing_Interface	9	Type: Unsigned8 Classification: Static Accessibilité: Read/write Valeurs nommées: 0: DL; 1: dorsale Cet attribut se rapporte à l'élément correspondant en Tableau 208 et est utilisé pour la configuration de la NL du nouvel appareil afin d'utiliser le contrat affecté au nouvel appareil

Le nouvel appareil doit utiliser les informations de DL fournies dans la DPDU d'annonce pour prendre en charge ce contrat. Après que l'appareil a rejoint le réseau, le gestionnaire de système doit accéder aux attributs de DLMO pertinents dans l'appareil pour modifier ces informations de DL selon le cas. Plus d'informations relatives à ces informations de DL et aux attributs de DLMO sont fournies en 9.1.14.

Si le nouvel appareil n'est pas autorisé par le gestionnaire de sécurité à rejoindre le réseau, il convient que le gestionnaire de sécurité informe le gestionnaire de système pour libérer ce contrat et les ressources réseau associées. Lorsqu'il en a ainsi reçu notification, le gestionnaire de système doit libérer le contrat et les ressources réseau associées d'un tel appareil qui n'est pas autorisé à rejoindre le réseau.

6.3.11.2.7 Qualité de service

6.3.11.2.7.1 Généralités

Le contrat assigné par le gestionnaire de système à un processus d'application demandeur indique également la qualité du service (QoS) pour le service de communication fourni. L'établissement de contrat doit être utilisé pour atteindre cet accord de QoS entre le processus d'application demandeur et le gestionnaire de système.

Un processus d'application qui veut communiquer avec son homologue peut indiquer la QoS désirée pour cette communication dans sa demande de contrat. Les arguments d'entrée décrits dans le Tableau 27 peuvent être utilisés dans ce but.

Les arguments d'entrée Contract_Priority et Payload_Size peuvent être utilisés dans des demandes de contrat concernant les services de communication tant périodiques que non

périodiques. Les arguments d'entrée Requested_Period, Requested_Phase et Requested_Deadline sont pertinents pour les communications périodiques. Les arguments d'entrée Committed_Burst, Excess_Burst et Max_Send_Window_Size sont pertinents pour les communications non périodiques. L'argument d'entrée Reliability_And_PublishDoNotAutoRetransmit contient des informations relatives à la fiabilité souhaitée qui sont pertinentes pour les communications tant périodiques que non périodiques. Il indique également si le processus d'application souhaite réémettre d'anciennes données de communication périodique si de nouvelles données ne sont pas disponibles.

Dans la réponse de contrat, le gestionnaire de système doit indiquer le niveau de QoS fourni pour le service de communication assigné. Les arguments de sortie décrits dans le Tableau 27 correspondant aux arguments d'entrée susmentionnés doivent être utilisés dans ce but.

6.3.11.2.7.2 Négociabilité de contrat

Une source qui envoie une demande de contrat doit également indiquer si, oui ou non, le service et la QoS de communication demandés sont négociables, c'est-à-dire, si, oui ou non, le gestionnaire de système peut assigner un contrat qui fournit un service et une QoS de communication différents de ceux demandés s'il ne peut pas prendre en charge la demande en l'état et si, oui ou non, le gestionnaire de système peut révoquer le contrat s'il y a lieu. Les arguments d'entrée Contract_Negotiability doivent être utilisés dans ce but.

Le Tableau 27 contient les arguments qui sont nécessaires pour la négociation du contrat entre la source et le gestionnaire de système. Si le gestionnaire de système n'est pas capable de prendre en charge une demande de contrat, il peut choisir de fournir des conseils de négociation de contrat. De tels conseils doivent être fournis en utilisant les arguments de sortie dans le Tableau 27 qui commencent par le mot "supportable", par exemple, Supportable_Contract_Priority.

Si le gestionnaire de système n'est pas capable de prendre en charge la demande de contrat au moment où elle a été reçue, mais il prévoit d'être capable de prendre en charge une telle demande dans le futur, il peut indiquer cela en utilisant l'argument de sortie Retry_Backoff_Time.

6.3.11.2.7.3 Priorités de contrat et priorités de message

Deux niveaux de priorité doivent être pris en charge dans le système, la priorité de contrat et la priorité de message.

La priorité de contrat doit établir une priorité de base pour tous les messages envoyés en utilisant ce contrat. Quatre priorités de contrat doivent être prises en charge en utilisant 2 bits comme suit:

- 0b11, Network control (commande réseau). La commande réseau peut être utilisée pour la gestion critique du réseau par le gestionnaire de système.
- 0b10, Real time buffer (tampon temps réel). Le tampon temps réel peut être utilisé pour les communications périodiques dans lesquelles le tampon de messages est écrasé en écriture chaque fois qu'un message plus récent est généré.
- 0b01, Real time sequential (temps réel séquentiel). Le temps réel séquentiel peut être utilisé pour les applications telles que la voix ou la vidéo qui ont besoin d'une livraison séquentielle des messages.
- 0b00, Best effort queued (meilleur effort mis en file d'attente). Le meilleur effort mis en file d'attente peut être utilisé pour les communications client/serveur.

La priorité de message doit établir une priorité au sein d'un contrat. Deux priorités de message doivent être prises en charge en utilisant 1 bit, low = 0 et high = 1 Un autre 1 bit est réservé pour de futures versions de la présente norme et doit être mis à 0.

La priorité de contrat doit être spécifiée par l'application, au cours du temps d'établissement de contrat, dans sa demande de contrat. Elle peut être utilisée par le gestionnaire de système pour établir les chemins préférentiels pour les contrats de haute priorité et pour équilibrer la charge du réseau. Le gestionnaire de système doit acheminer la priorité assignée du contrat vers la source dans la réponse de contrat.

La priorité de message doit être fournie par l'application pour chaque message envoyé en descendant la suite de protocoles. Dans la source, la priorité de message doit descendre la suite de protocoles. La priorité de contrat doit être ajoutée au niveau de la NL. La priorité de contrat doit prévaloir sur la priorité de message.

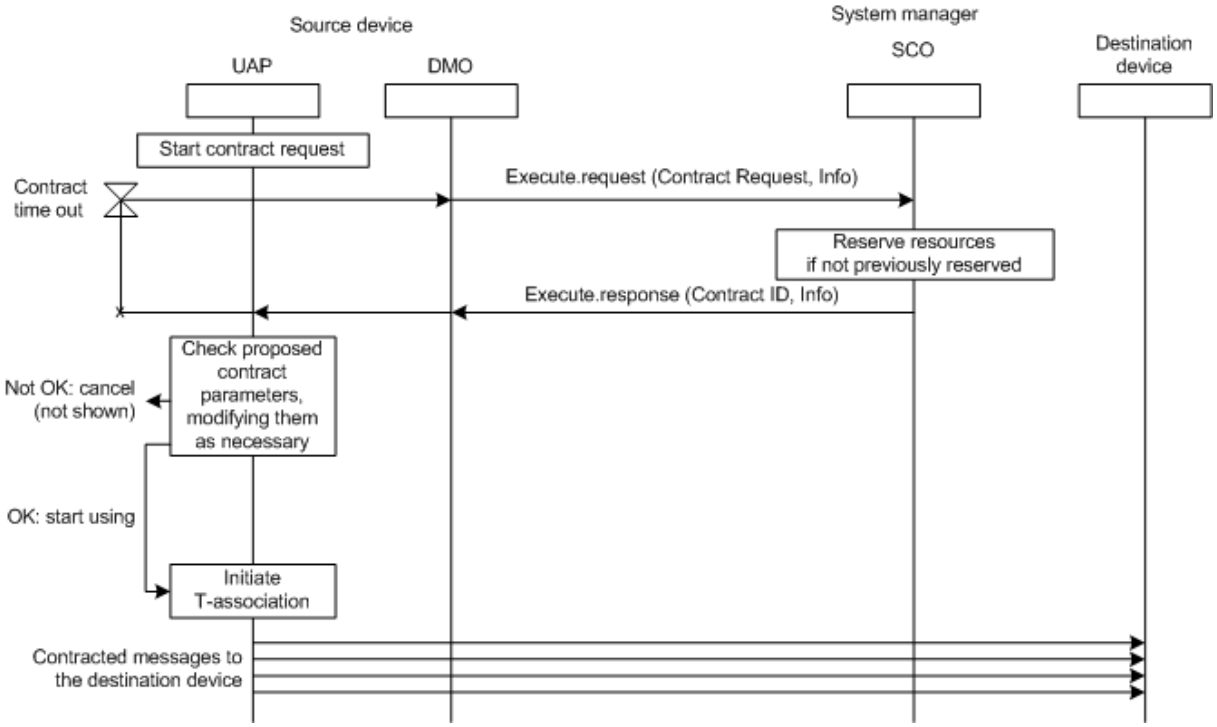
La priorité combinée de contrat/message doit être utilisée pour résoudre la compétition pour des ressources limitées lorsque ces messages sont transmis à travers le réseau. La DL doit utiliser ces informations pour conduire des décisions de mise en file d'attente lorsqu'elle transmet des messages sur le sous-réseau D. Elles ne doivent être incluses que dans l'en-tête de la DL. Quand le message est envoyé sur une dorsale, la priorité doit être incluse dans les en-têtes de réseau. La NL doit utiliser une priorité pour conduire des décisions de mise en file d'attente sur une dorsale.

6.3.11.2.7.4 Arguments relatifs à la phase

L'argument d'entrée Requested_Phase doit être utilisé par le processus d'application sollicitant le contrat pour demander une phase qui est le décalage temporel par rapport au début d'une période. Le décalage temporel est exprimé sous la forme d'un pourcentage du temps au sein d'une période. Toutes les périodes doivent être calculées de sorte que leurs temps de début soient synchrones du début du temps TAI. Les applications peuvent utiliser l'argument Requested_Phase pour obtenir une exécution en boucle synchronisée et distribuée avec une latence minimale et une gigue bornée. La temporisation exacte de la phase telle qu'elle se rapporte à la DL est spécifiée par le nombre de liaisons, qui est décrit en 9.4.3.7.

6.3.11.2.8 Diagramme de séquence de message d'établissement de contrat

La Figure 30 montre un exemple d'un diagramme de séquence de message pour l'établissement d'un contrat. Cet exemple n'implique pas de temporisations et l'appareil de source accepte le contrat établi par le gestionnaire de système même si ce contrat fournit un service de communication différent de celui demandé.



Légende

Anglais	Français
Source device	Appareil source
System manager	Gestionnaire de système
Destination device	Appareil de destination
Start contract request	Démarrer demande de contrat
Contract time out	Temporisation contrat
Execute.request (Contract Request, Info)	Execute.request (Contract Request, Info)
Reserve resources if not previously reserved	Réserver des ressources si elles ne l'ont pas été
Execute.response (Contract ID, Info)	Execute.response (Contract ID, Info)
Not OK: cancel (not shown)	Pas OK; annuler (pas montré)
Check proposed contract parameters, modifying them as necessary	Vérifier les paramètres contractuels proposés, les modifier si nécessaire
OK: start using	OK: commencer à utiliser
Initiate T-association	Initier association T
Contracted messages to the destination device	Messages contractés vers l'appareil de destination

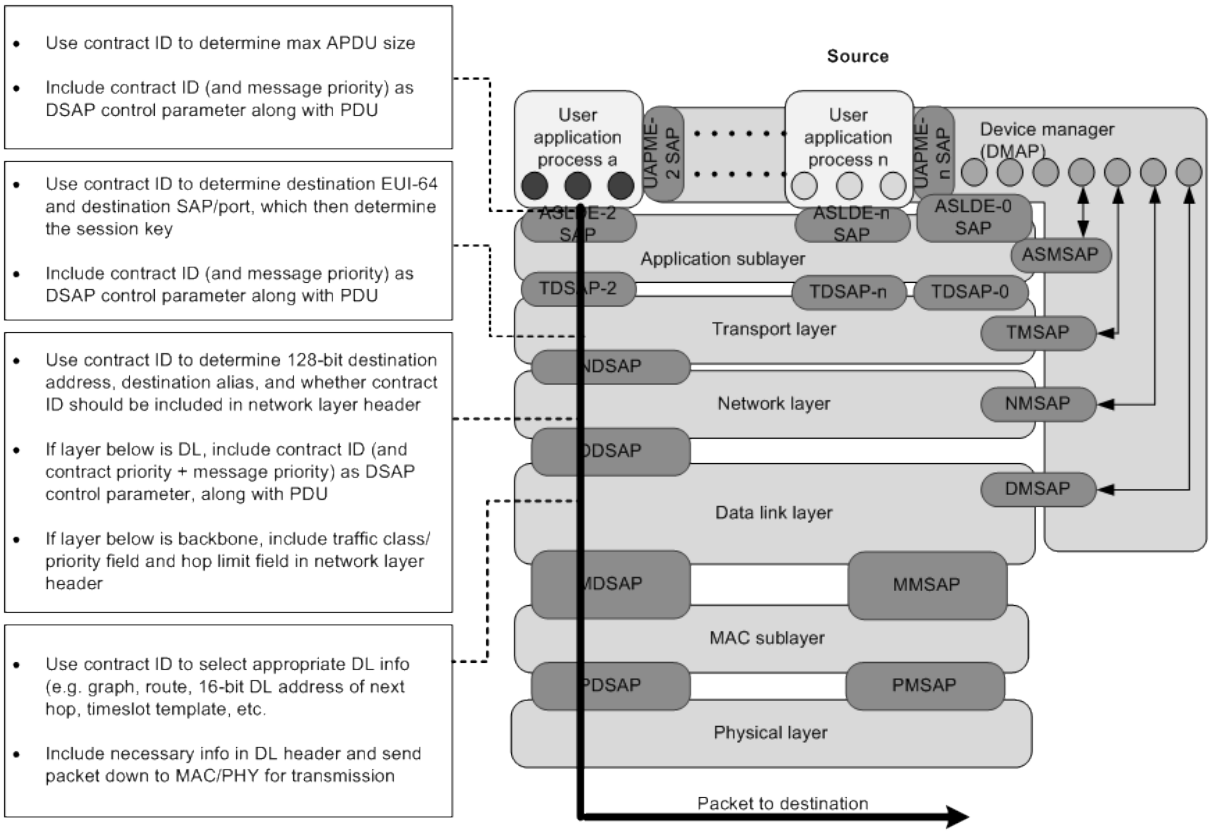
Figure 30 – Exemple d'établissement de contrat

6.3.11.2.9 Utilisation d'identificateur de contrat

6.3.11.2.9.1 Généralités

L'ID de contrat doit être fourni par le gestionnaire de système à la source. Le processus d'application demandant un contrat doit récupérer l'ID de contrat assigné et doit l'utiliser pour envoyer des unités de données de protocole en descendant la suite de protocoles. Comme cela a été décrit précédemment, chaque couche de la source est configurée pour traiter de PDU de couche supérieure qui sont accompagnées de l'ID de contrat, qui est passé comme un paramètre de commande de DSAP.

La Figure 31 montre comment l'ID de contrat doit être utilisé lorsque l'unité de données descend la suite de protocoles de la source.



Légende

Anglais	Français
Source	Source
User application process a	Processus d'application utilisateur a
User application process n	Processus d'application utilisateur n
Device manager (DMAP)	Gestionnaire d'appareil (DMAP)
Application sub-layer	Sous-couche application
Transport layer	Couche transport
Network layer	Couche réseau
Data link layer	Couche liaison de données
MAC sub-layer	Sous-couche MAC
Physical layer	Couche physique
Packet to destination	Paquet à la destination
Use contract ID to determine max APDU size	Utiliser l'ID de contrat pour déterminer la taille d'APDU maximale
Include contract ID (and message priority) as DSAP control parameter along with PDU	Inclure l'ID de contrat (et la priorité de message) comme paramètre de commande de DSAP, avec la PDU
Use contract ID to determine destination EUI-64 and destination SAP/port, which then determine the session key	Utiliser l'ID de contrat pour déterminer l'EUI-64 de destination et le port/SAP de destination, qui déterminent alors la clé de session
Include contract ID (and message priority) as DSAP control parameter along with PDU	Inclure l'ID de contrat (et la priorité de message) comme paramètre de commande de DSAP, avec la PDU
Use contract ID to determine 128-bit destination address, destination alias, and whether contract ID should be included in network layer header	Utiliser l'ID de contrat pour déterminer l'adresse de destination de 128 bits, le pseudonyme de destination et si, oui ou non, il convient d'inclure l'ID de contrat dans l'en-tête de couche réseau

Anglais	Français
If layer below is DL, include contract ID (and contract priority + message priority) as DSAP control parameter, along with PDU	Si la couche en dessous est DL, inclure l'ID de contrat (et la priorité du contrat + la priorité de message) comme paramètre de commande de DASP, avec la PDU
If layer below is backbone, include traffic class/ priority field and hop limit field in network layer header	Si la couche en dessous est la dorsale, inclure le champ classe de trafic/priorité et le champ limite de sauts dans les en-têtes de couche réseau
Use contract ID to select appropriate DL info (e.g. graph, route, 16-bit DL address of next hop, timeslot template, etc.)	Utiliser l'ID de contrat pour sélectionner les info adéquates de DL (par exemple: graphe, chemin, adresse DL de 18 bits du prochain saut, le modèle d'intervalle de temps, etc.)
Include necessary info in DL header and send packet down to MAC/PHY for transmission	Inclure les info nécessaires dans l'en-tête de DL et descendre le paquet vers la MAC/PHY pour émission

Figure 31 – Utilisation de l'ID de contrat dans la source

6.3.11.2.9.2 Utilisation de l'identificateur de contrat dans les routeurs dorsaux intermédiaires

L'inclusion de l'ID de contrat dans l'en-tête de réseau de la NPDU par la source doit être configurée par le gestionnaire de système. Si le chemin de communication de la source vers la destination passe par la dorsale, alors le gestionnaire de système doit informer la source d'inclure l'ID de contrat dans son en-tête de réseau. Plus de détails sont fournis en 10.5.3.

6.3.11.2.9.3 Relation entre contrats et alertes

L'accès au DMAP est limité au SMAP qui réside dans le gestionnaire de système. En contradiction avec ce principe général, les maîtres d'alertes sont autorisés à accéder à l'objet ARMO présent dans le DMAP. L'accès de DMAP par les maîtres d'alertes doit être limité à l'ARMO, à moins que le maître d'alertes n'utilise la session DMAP-SMAP établie lorsque l'appareil avait rejoint le réseau. L'ARMO dans le DMAP doit émettre les alertes qui appartiennent aux différentes catégories alertes vers les maîtres d'alertes respectifs qui sont décrits dans le 6.2.7.2. Si ces maîtres d'alertes sont des appareils différents avec leurs propres adresses IPv6Address uniques, l'ARMO doit avoir un contrat séparé avec chacun d'eux pour communiquer les alertes. L'ARMO dans l'appareil demande ces contrats au gestionnaire de système par l'intermédiaire du DMO dans l'appareil.

6.3.11.2.10 Résiliation, désactivation et réactivation de contrat

6.3.11.2.10.1 Généralités

Les contrats peuvent être résiliés lorsque le besoin de communication qui avait établi le contrat a été satisfait. Les contrats peuvent également être résiliés lorsque la source ou la destination n'est plus disponible.

Lorsqu'il y a résiliation de contrat, le SCO doit informer le DMO de la source, si la source est encore disponible. Le DMO à son tour informe le processus d'application qui utilisait ce contrat.

Lorsqu'il y a résiliation de contrat, le SCO peut également libérer les ressources réseau qui étaient allouées pour prendre en charge le contrat. En outre, les informations relatives à la sécurité, y compris les clés T entre la source et la destination, peuvent également être supprimées par le gestionnaire de sécurité selon les interactions avec le gestionnaire de système.

Un contrat peut également être désactivé si le besoin de communication est censé être suspendu pendant une période. Le contrat peut être réactivé lorsque le besoin de communication reprend.

6.3.11.2.10.2 Résiliation de contrat lorsqu'un appareil quitte le réseau ou n'est plus disponible

Lorsque le gestionnaire de système détermine qu'un appareil ne fait plus partie du réseau, il doit résilier tous contrats associés à cet appareil et libérer les ressources réseau qui étaient allouées pour prendre en charge ces contrats. Le gestionnaire de système peut utiliser des informations issues d'autres appareils dans le voisinage de cet appareil pour décider que cet appareil ne fait plus partie du réseau. Le gestionnaire de système peut lire l'attribut de dlmo.Neighbor (décrit en 9.4.3.4) de ces appareils voisins pour prendre cette décision.

Lorsqu'un appareil qui a l'attribut de DMO Non_Volatile_Memory_Capability = 1 perd la connectivité réseau/cycles de puissance ou passe par un redémarrage à chaud pour une raison quelconque, il doit maintenir toutes les informations nécessaires relatives aux contrats telles que décrites en 6.3.9.4.2. Ainsi, cet appareil peut reprendre le fonctionnement normal dès qu'il aura rétabli la synchronisation du temps avec le réseau. L'appareil est censé rétablir la synchronisation du temps en se mettant à l'écoute pour détecter des annonces ou en sollicitant des annonces.

Si le gestionnaire de système avait résilié tous les contrats de cet appareil alors que l'appareil ne faisait pas partie du réseau, l'appareil est censé ne pas réussir à reprendre le fonctionnement normal et il est donc censé exécuter un cycle restartAsProvisioned. Cet appareil doit maintenir toutes les informations qui lui avaient été fournies pendant l'étape de configuration avant qu'il n'ait rejoint le réseau pour la première fois ainsi que toutes les informations constantes et statiques présentes dans les UAP.

Lorsqu'un appareil qui a l'attribut de DMO Non_Volatile_Memory_Capability = 0 perd la connectivité réseau, procède à une mise sous/hors tension ou subit un cycle restartAsProvisioned pour une raison quelconque, il est censé répéter le processus de rattachement en utilisant les informations qui lui avaient été fournies pendant l'étape de configuration avant de rejoindre le réseau pour la première fois. Cet appareil doit également retenir toutes les informations constantes et statiques présentes dans les UAP.

Le DMO d'un appareil qui se réinitialise à l'état d'usine par défaut ou subit un cycle restartAsProvisioned doit résilier tous ses contrats en utilisant la méthode définie dans le Tableau 27 avant de se réinitialiser ou de redémarrer.

6.3.11.2.10.3 Résiliation de contrat lorsque la clé T est résiliée

Le gestionnaire de système peut résilier un contrat d'un appareil particulier s'il est informé par le gestionnaire de sécurité qu'une clé T correspondante de l'appareil en question a été résiliée. Tout contrat de l'appareil qui emploie l'accès T correspondant à cette clé T particulière de T peut être résilié.

6.3.11.2.10.4 Appareils qui peuvent résilier, désactiver et réactiver des contrats

Seuls la source ou le gestionnaire de système doit avoir la capacité de résilier un contrat existant de la source.

Seule la source doit avoir la capacité de désactiver et de réactiver un contrat existant de la source.

6.3.11.2.10.5 Arguments de demande et de réponse de résiliation, de désactivation et de réactivation de contrat

Si la source décide de résilier un contrat, elle doit envoyer une demande de résiliation de contrat au SCO. Le SCO doit alors envoyer la réponse en retour à la source l'informant que le contrat a été résilié. La demande doit être une Execute.Request au SCO avec l'ID de contrat comme l'un des arguments d'entrée, et la réponse doit être une Execute.Response avec le statut comme l'argument de sortie. Cela est décrit au Tableau 32.

Si la source décide de désactiver/réactiver un contrat, elle doit envoyer une demande de désactivation/réactivation de contrat au SCO. Le SCO doit alors envoyer la réponse en retour à la source l'informant que le contrat a été désactivé/réactivé. La demande doit être une Execute.Request au SCO avec l'ID de contrat comme l'un des arguments d'entrée, et la réponse doit être une Execute.Response avec le statut comme l'argument de sortie. Cela est décrit dans le Tableau 32.

Tableau 32 – Méthode du SCO pour la résiliation, la désactivation et la réactivation de contrat

<table>
<tr><th colspan="5">Nom du type d'objet normalisé: System communication configuration object (SCO, objet de configuration de communications système)</th></tr>
<tr><th colspan="5">Identificateur du type d'objet normalisé: 102</th></tr>
<tr><th>Nom de la méthode</th><th>ID de la méthode</th><th colspan="3">Description de la méthode</th></tr>
<tr><td rowspan="8">Contract_Termination
_Deactivation_Reactivation</td><td>2</td><td colspan="3">Méthode permettant de résilier, de désactiver ou de réactiver un contrat</td></tr>
<tr><td colspan="4">Arguments d'entrée</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td>1</td><td>Contract ID</td><td>Unsigned16</td><td>ID du contrat résilié, désactivé ou réactivé</td></tr>
<tr><td>2</td><td>Operation</td><td>Unsigned8</td><td>Valeurs nommées:
0: résiliation du contrat;
1: désactivation du contrat;
2: réactivation du contrat</td></tr>
<tr><td colspan="4">Arguments de sortie</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td>1</td><td>Erreur</td><td>Unsigned8</td><td>Valeurs nommées:
0: réussite;
> 0: échec</td></tr>
</table>

Si le gestionnaire de système décide de résilier un contrat, il doit envoyer une commande de résiliation de contrat avec l'ID de contrat au DMO de la source. Le DMO doit alors retourner une réponse avec le statut.

La méthode du DMO permettant de résilier un contrat existant est décrite dans le Tableau 33.

Tableau 33 – Méthode du DMO pour résilier un contrat

Nom du type d'objet normalisé: Device management object (DMO, objet de gestion d'appareil)		
Identificateur du type d'objet normalisé: 127		
Nom de la méthode	**ID de la méthode**	**Description de la méthode**
Contract_Terminated	1	Méthode permettant de résilier un contrat existant dans l'attribut Contracts_Table dans le Tableau 10. Cette méthode utilise le modèle de méthode Delete_Row défini dans le Tableau J.5 avec les arguments suivants: Attribute_ID: 26 (Contracts_Table) Index_1: 1 (Contract_ID)

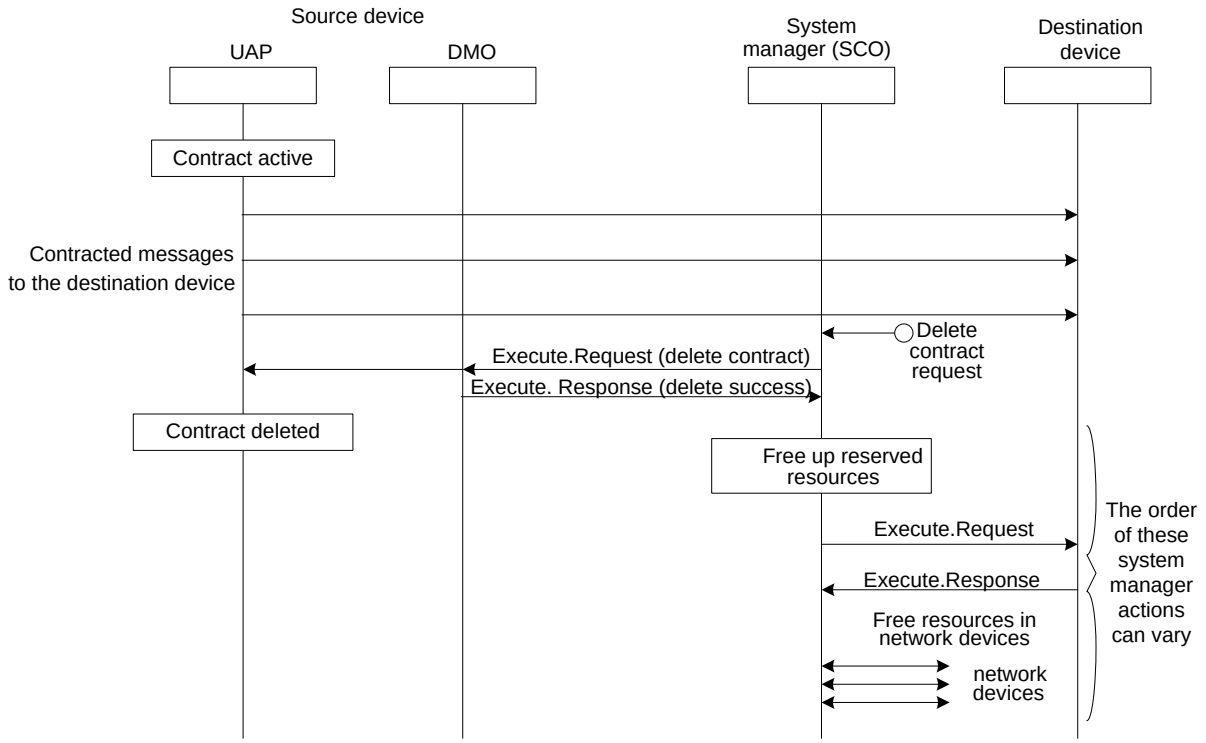
6.3.11.2.10.6 Configuration de suites de protocoles

Lorsque le SCO résilie un contrat, en plus d'informer la source au sujet de la résiliation, il peut également libérer les ressources réseau qui avaient été allouées dans la source, la destination, et les appareils intermédiaires. Des procédures semblables à celles utilisées pour la configuration de suites de protocoles pendant l'établissement de contrat (voir 6.3.11.2.6) peuvent être utilisées par le SCO pour libérer ces ressources réseau.

Le SCO informe le gestionnaire de sécurité, par l'intermédiaire du PSMO, de la résiliation du contrat. Le gestionnaire de sécurité peut décider de supprimer la clé T qui a été assignée pour la communication entre la source et la destination. Dans ce cas, le gestionnaire de sécurité doit envoyer les messages d'effacement de clé T à la source et la destination par l'intermédiaire du PSMO.

6.3.11.2.10.7 Diagramme de séquence de message de résiliation de contrat

La Figure 32 montre le diagramme de séquence de message pour la résiliation d'un contrat initiée par le gestionnaire de système.



Légende

Anglais	Français
Source device	Appareil source
System manager (SCO)	Gestionnaire de système (SCO)
Destination device	Appareil de destination
Contract active	Contrat actif
Contracted messages to the destination device	Messages contractés envoyé vers appareil de destination
Execute.Request (delete contract)	Execute.Request (supprimer contrat)
Execute. Response (delete success)	Execute.Response (supprimer contrat)
Delete contract request	Demande de suppression de contrat
Contract deleted	Contrat supprimé
Free up reserved resources	Libérer des ressources réservées

Anglais	Français
Execute.Request	Execute.Request
Execute.Response	Execute.Response
Free resources in network devices	Libérer des ressources dans les appareils de réseau
Network devices	Appareils de réseau
The order of these system manager actions can vary	L'ordre de ces actions du gestionnaire de système peut varier

Figure 32 – Résiliation de contrat

6.3.11.2.11 Maintenance et modification de contrat

6.3.11.2.11.1 Généralités

Le SCO a besoin de maintenir les contrats établis en s'assurant que les ressources réseau allouées sont disponibles dans des conditions normales. Si les ressources réseau allouées deviennent indisponibles, le SCO peut choisir d'allouer des ressources réseau alternatives afin de continuer à maintenir le contrat établi.

Un contrat peut être modifié si le besoin de communication de l'application correspondante (prise en charge par ce contrat) change. Un contrat peut également être modifié si le gestionnaire de système décide de changer les ressources réseau allouées pour le contrat.

Les modifications de contrat s'inscrivent dans deux catégories:

- modifications conduisant à une réduction des ressources réseau allouées; et
- modifications conduisant à un changement ou à une augmentation des ressources réseau allouées.

Pour les modifications de contrat à l'initiative de l'application, ces deux catégories suivent des étapes légèrement différentes.

Les modifications de contrat qui conduisent à une réduction des ressources réseau allouées peuvent avoir un effet immédiat, à savoir que la source peut commencer à utiliser la configuration de suite de protocoles du contrat modifié dès qu'elle reçoit la réponse accompagnée de ces informations de configuration en provenance du SCO si cette réponse l'indique dans l'argument de sortie Response_Code.

Les modifications de contrat qui conduisent à une augmentation ou à un changement des ressources réseau allouées ne doivent pas avoir un effet immédiat, à savoir que la source ne doit pas commencer à utiliser la configuration de suite de protocoles du contrat modifié dès qu'elle reçoit la réponse accompagnée de ces informations de configuration en provenance du SCO. La raison en est que le SCO a encore besoin d'augmenter ou de changer l'allocation des ressources réseau. La réponse issue du SCO doit inclure un argument de sortie Activation_Time qui indique à la source l'instant auquel elle peut commencer à utiliser la nouvelle configuration de suite de protocoles. Cela conduit à un effet différé.

6.3.11.2.11.2 Appareils qui peuvent modifier des contrats

Seule la source ou seul le gestionnaire de système doit avoir la capacité de modifier un contrat existant de cette source.

6.3.11.2.11.3 Arguments de demande et de réponse de modification de contrat

Si la source décide de modifier un contrat, elle doit envoyer une demande de modification de contrat au SCO. Le SCO doit alors envoyer une réponse en retour à la source l'informant que le contrat a été modifié. La demande doit être une Execute.Request au SCO, et la réponse

doit être un message Execute.Response. Les arguments d'entrée et de sortie sont fournis dans le Tableau 27. Le SCO peut également communiquer avec les appareils pertinents pour allouer ou désallouer les ressources réseau nécessaires.

Si le gestionnaire de système décide de modifier un contrat, il doit envoyer une commande de modification de contrat au DMO de la source en utilisant la méthode Modify_Contract. Le DMO doit alors envoyer une réponse en retour avec le statut. Le SCO peut également communiquer avec les appareils pertinents pour allouer ou désallouer les ressources réseau nécessaires.

La méthode du DMO permettant de modifier un contrat existant est décrite dans le Tableau 34.

Tableau 34 – Méthode du DMO pour modifier un contrat

Nom du type d'objet normalisé: Device management object (DMO, objet de gestion d'appareil)		
Identificateur du type d'objet normalisé: 127		
Nom de la méthode	**ID de la méthode**	**Description de la méthode**
Contract_Modified	2	Méthode permettant de modifier un contrat existant dans l'attribut Contracts_Table dans le Tableau 10. Cette méthode utilise le modèle de méthode Write_Row défini dans le Tableau J.3 avec les arguments suivants: Attribute_ID: 26 (Contracts_Table) Index_1: 1 (Contract_ID)

6.3.11.2.11.4 Renouvellement de contrat

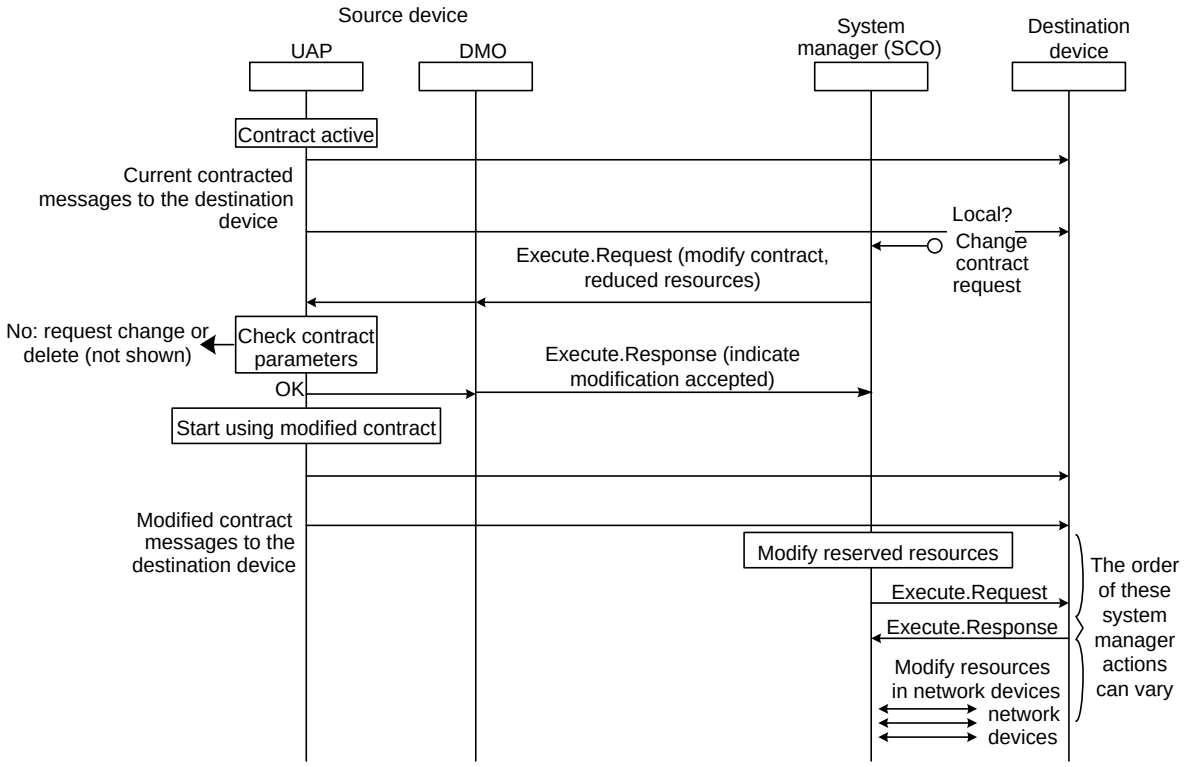
Le renouvellement de contrat est l'équivalent d'une simple modification de contrat, seul l'argument d'entrée Contract_Expiration_Time étant mis à jour et tous les autres arguments d'entrée restant les mêmes que ceux contenus dans la demande de contrat d'origine.

6.3.11.2.11.5 Configuration de suites de protocoles

Comme partie intégrante de la modification de contrat, le SCO doit configurer / reconfigurer les appareils nécessaires dans le réseau en fournissant à chacun de ceux-ci les configurations nécessaires de suites de protocoles. Cela doit inclure la reconfiguration de la destination et de la source. Des procédures semblables à celles utilisées pour la configuration de suites de protocoles pendant l'établissement de contrat (voir 6.3.11.2.6) peuvent être utilisées par le SCO dans ce but.

6.3.11.2.11.6 Diagramme de séquence de message de modification de contrat

La Figure 33 montre le diagramme de séquence de message pour modifier un contrat avec effet immédiat.



Légende

Anglais	Français
Source device	Appareil de source
System manager (SCO)	Gestionnaire de système (SCO)
Destination device	Appareil de destination
Contract active	Contrat actif
Current contracted messages to the destination device	Message contractés courant envoyés vers appareils de destination
Execute.Request (modify contract, reduced resources)	Execute.Request (modifier contrat, ressources réduites)
Local? Change contract request	Local? Changer demande de contrat
No: request change or delete (not shown)	Non: demander de changer ou de supprimer (pas montré)
Check contract parameters	Vérifier les paramètres du contrat
Execute.Response (indicate modification accepted)	Execute.Response (indiquer modification acceptée)
Ok	Ok
Start using modified contract	Commencer à utiliser contrat modifié
Modified contract messages to the destination device	Message de contrat modifié envoyés vers appareils de destination
Modify reserved resources	Modifier ressources réservées
The order of these system manager actions can vary	L'ordre de ces actions du gestionnaire de système peut varier
Execute.Request	Execute.Request
Execute.Response	Execute.Response
Modify resources in network devices	Modifier ressources dans les appareils de réseau
Network devices	Appareil de réseau

Figure 33 – Modification de contrat avec effet immédiat

6.3.11.2.11.7 Modification de contrat et mises à jour de clé T

Les mises à jour de clé T ne sont pas traitées comme étant des modifications de contrat. De telles mises à jour de clés doivent être envoyées du gestionnaire de sécurité, en passant par l'objet proxy de gestion de sécurité (PSMO) dans le gestionnaire de système, vers les appareils pertinents qui possèdent la session correspondante.

6.3.11.2.12 Scénarios d'échec de contrat

Le Tableau 27 contient des arguments de sortie pour les scénarios d'échec dans lesquels le gestionnaire de système n'est pas capable de prendre en charge la demande de contrat. De tels échecs peuvent se produire si le service de communication demandé ne peut pas être pris en charge du tout, s'il ne peut pas être pris en charge en raison d'une condition temporaire, ou s'il ne peut pas être pris en charge si la demande n'est pas envoyée de nouveau par la source avec des arguments négociés à la baisse. Dans ces cas, le gestionnaire de système peut choisir d'inclure des arguments de sortie dans la réponse qui fournissent quelques conseils à la source. Ceux-ci comprennent Retry_Backoff_Time et Negotiation_Guidance.

6.3.12 Gestion de redondance

Bien que la présente norme incorpore des caractéristiques qui permettent la prise en charge de la connexion sans fil tant simplex que totalement redondante des appareils de terrain vers un réseau dorsal d'installation, la gestion de cette redondance n'est pas spécifiée dans la présente norme.

Le gestionnaire de système est censé être capable de configurer la redondance de chemin dans le sous-réseau D par l'intermédiaire des routeurs de terrain. Les appareils de terrain, y compris les routeurs de terrain, peuvent être configurés pour communiquer avec les routeurs dorsaux redondants.

La redondance au niveau appareil qui requiert une synchronisation entre les appareils redondants pour maintenir les informations d'état est autorisée, mais n'est pas spécifiée dans la présente norme.

6.3.13 Protocoles de gestion de système

La communication relative à la gestion entre appareils conformes à la présente norme et le gestionnaire de système doit être accomplie par l'intermédiaire de la messagerie normalisée de sous-couche d'application, comme cela est décrit en 12.12.

6.3.14 Politiques de gestion et administration de politiques

Les politiques de gestion et l'administration de politiques ne sont pas spécifiées par la présente norme. Une politique par défaut peut être établie pour rendre toutes les informations relatives à un appareil disponibles pour le gestionnaire de système (avec la sécurité appropriée). Des informations de vue d'ensemble peuvent être rendues disponibles à l'extérieur du réseau (par exemple, le réseau fonctionne dans des limites nominales).

6.3.15 Interaction opérationnelle avec le personnel d'exploitation et de maintenance d'installation

Alors que l'appareil mettant en œuvre le gestionnaire de système peut avoir une interface qui permet au personnel d'exploitation et de maintenance d'installation d'observer et commander les performances du réseau et des appareils, cette interface n'est ni obligatoire ni spécifiée par la présente norme.

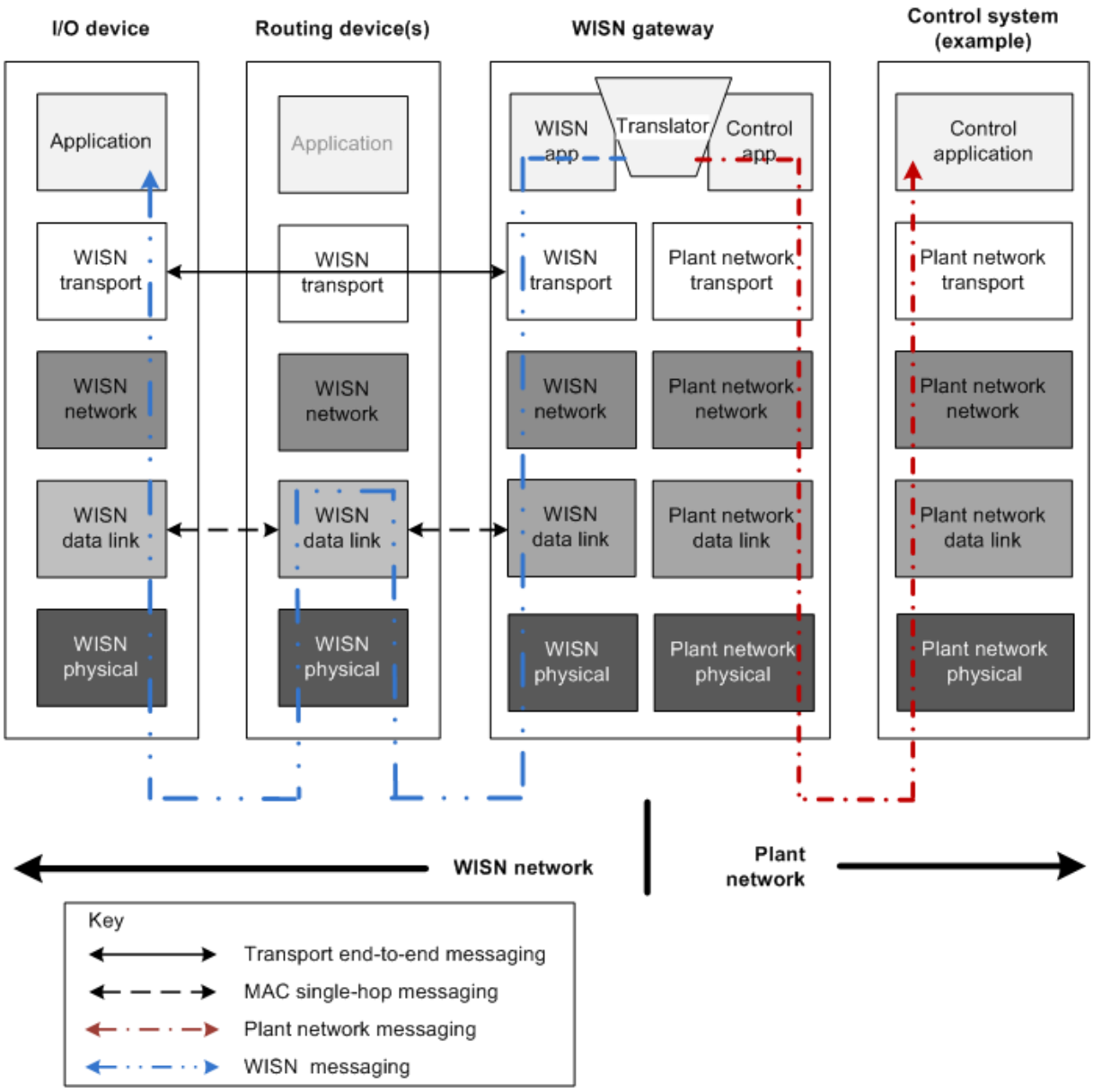
7 Sécurité

7.1 Généralités

L'Article 7 décrit la fonctionnalité des composants de sécurité, son interface avec la DLE et la TLE, et la protection des données en transit. Il décrit également le rôle de gestionnaire de sécurité.

Le point central de l'Article 7 est de fournir la sécurité d'émission et les aspects de sécurité connexes, y compris le processus de rattachement, l'établissement de session, les mises à jour de clés, et les politiques associées. La présente norme ne traite pas des autres types de sécurité, tels que la sécurité des données au repos ou la sécurité d'appareil physique.

Les messages spécifiques qui sont protégés sont des DPDU à saut unique (saut par saut), les TPDU de transport de bout en bout et les structures de données de gestion de sécurité lorsqu'elles sont transportées dans des APDU. Un flot de données en régime établi utilisant des DPDU et des TPDU qui peuvent être protégées est décrit dans les grandes lignes à la Figure 34. Les points d'extrémité de TLE d'une association de sécurité T sont définis par les appareils de points d'extrémité ainsi que l'application finale.



Légende

Anglais	Français
I/O device	Appareil E/S
Application	Application
WISN transport	Transport de WISN
WISN network	Réseau de WISN
WISN data link	Liaison de données de WISN
WISN physical	Physique de WISN
Routing device(s)	Appareil(s) de routage
WISN gateway	Passerelle WISN
WISN app	App WISN
Translator	Convertisseur
Control app	App de commande
Plant network transport	Transport de réseau d'installation
Plant network network	Réseau de réseau d'installation
Plant network data link	Liaison de données de réseau d'installation

Anglais	Français
Plant network physical	Physique de réseau d'installation
Control system (example)	Système de commande (exemple)
Control application	Application de commande
Plant network	Réseau d'installation
Transport end-to-end messaging	Message de bout en bout de transport
MAC single-hop messaging	Messagerie à un seul saut de MAC
Plant network messaging	Messagerie de réseau d'installation
WISN messaging	Messagerie de WISN

Figure 34 – Exemples de portée de DPDU et de TPDU

7.2 Services de sécurité

7.2.1 Vue d'ensemble

Les services de sécurité dans la présente norme sont sélectionnés par la politique. La politique est distribuée avec chaque support cryptographique, permettant une application de politique focalisée. Une seule clé étant utilisée à un instant donné à la DL, à l'exception d'une brève période de commutation de clé, le sous-réseau entier est soumis aux mêmes politiques à la DL. Le gestionnaire de sécurité commande les politiques pour tous les supports cryptographiques qu'il produit.

Les appareils avec les authentifiants appropriés participent aux communications sécurisées avec d'autres appareils par l'utilisation d'une clé symétrique secrète partagée qui est utilisée pour authentifier et, par configuration de sécurité, pour chiffrer leurs messages entre eux.

NOTE 1 Bien que l'authentification implique l'utilisation d'une primitive de chiffrement, elle ne conduit pas à la confidentialité du contenu des messages; un processus de chiffrement (passe) distinct est exigé pour la confidentialité du contenu des messages.

Les services de sécurité sont appliqués au fond de la pile de protocoles de communication, houblon-saut par saut à la DL, et au sommet de la pile de protocoles de communication, bout en bout à la TL. Les services de gestion de sécurité sont également utilisés par l'AL pour le processus de rattachement, la distribution de clé et la gestion de session. Lorsque des clés secrètes sont utilisées, la sécurité de DL protège contre les attaquants qui sont à l'extérieur du système et ne partagent pas de secrets du système, alors que la sécurité de TL protège contre les attaquants qui peuvent être sur le chemin de réseau entre la source et la destination.

Dans les deux cas, une clé de données symétrique (également appelée clé T), partagée parmi des communicants prévus, est utilisée pour ajouter un contrôle d'intégrité de message (MIC) codé et dur du point de vue cryptographique à la PDU et, quand cela est spécifié, pour fournir la confidentialité (par chiffrement) de la charge utile de la PDU. Les attaquants qui ne partagent pas la clé ne peuvent pas modifier le message sans une très forte probabilité de détection et ne peuvent déchiffrer aucune charge utile chiffrée.

L'opération de sécurité est basée sur un sens partagé du temps qui habituellement est aligné avec le temps TAI (voir 5.6). Les DLE et les TLE expéditrices s'authentifient à leurs homologues destinataires en utilisant le temps TAI nominal d'émission de DPDU et le temps approché de la création de TPDU.

Lorsque trois appareils ou plus partagent une clé secrète commune, l'authentification de la source n'est plus garantie au sein du groupe en raison de la clé symétrique partagée. Dans ce cas, l'authentification de la source au sein d'un groupe exige des mécanismes complexes; ainsi, l'authentification du nœud expéditeur spécifique (au sein du groupe de multidiffusion) n'est pas adressée.

Les composants de sécurité principaux des services fournis comprennent:

- l'autorisation de relations de communication sécurisées entre les entités;
- l'authenticité de message, assurant que les messages sont émis par un membre autorisé d'une relation de communications et qu'ils n'ont pas été modifiés alors qu'ils sont en transit entre l'émetteur et le destinataire par une entité à l'extérieur de la relation;
- l'assurance que la temporisation de la livraison et le réarrangement des messages n'excèdent pas les limites envisagées;
- la confidentialité des données qui cache le contenu (autre que la taille) parmi les charges utiles de message; et
- la protection contre les attaques par rejet malveillantes.

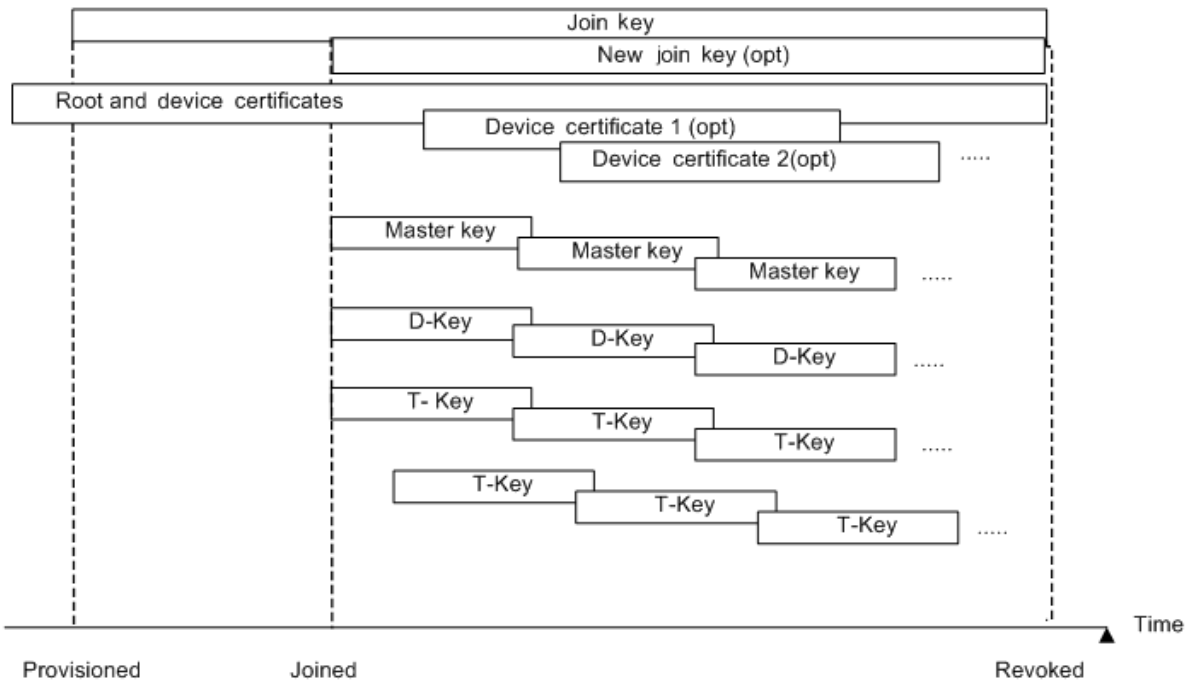
Diverses combinaisons de ces services sont fournies tant à une DLE qu'à une TLE. En outre, divers services cryptographiques sont disponibles pour être utilisés par le DSMO pour le processus de rattachement, l'établissement de session et la mise à jour de clé.

NOTE 2 La protection contre la compromission des frontières cryptographiques à l'intérieur du matériel des appareils conformes à la présente norme ne relève pas du domaine d'application de la présente norme. D'autres publications, y compris l'ISO/IEC 15408 et l'ISO/IEC 19790 (similaires à la série [US] NIST FIPS 140), traitent de ces questions. Les décisions de conformité sont laissées à ceux qui évaluent les appareils.

7.2.2 Clés

7.2.2.1 Généralités

Des clés symétriques sont utilisées pour le chiffrement et l'authentification des données (voir 7.3.2.5, 7.3.2.6, 7.3.3.8 et 7.3.3.9). Des clés asymétriques peuvent être utilisées pour le processus de rattachement, voir 7.4. Chaque clé est limitée dans le temps et peut être mise à jour. La Figure 35 montre les types de clés spécifiées par la présente norme et leurs durées de vie associées, y compris un certificat de sécurité à clés asymétriques (s'il en existe un).



Légende

Anglais	Français
Join key	Clé de rattachement
New Join key (opt)	Nouvelle clé de rattachement (facultative)

Anglais	Français
Root and device certificates	Certificats de racine et d'appareil
Device certificate 1 (opt)	Certificat d'appareil 1 (facultatif)
Device certificate 2 (opt)	Certificat d'appareil 2 (facultatif)
Master key	Clé principale
D-Key	Clé D
T-Key	Clé T
Provisioned	Configuré
Joined	Rattaché
Revoked	Révoqué
Time	Temps

Figure 35 – Clés et durées de vie associées

7.2.2.2 Clés symétriques

Toutes les clés symétriques WISN doivent être des valeurs à 128 bits. Les clés symétriques utilisées comprennent:

- Clé globale (global key): clé bien connue qui ne peut pas être utilisée pour garantir la moindre propriété de sécurité et qui n'expire jamais.
- K_open: clé globale utilisée comme clé de rattachement dans l'étape de configuration décrite en 13.3. La valeur réelle pour cette clé est 0x004F 0050 0045 004E 0000 0000 0000 0000, qui est la représentation de la chaîne Unicode de 16 octets terminée par un caractère nul "OPEN(null)(null)(null)(null)". L'identificateur CryptoKeyIdentifier pour cette clé est 1.
- K_global: clé globale utilisée comme clé de rattachement dans la phase de configuration et comme clé D dans la phase de rattachement. L'utilisation de cette clé dans la phase de configuration est décrite en 13.3. La valeur réelle pour cette clé est 0x0049 0053 0041 0020 0031 0030 0030 0000, qui est la représentation de la chaîne Unicode de 16 octets terminée par un caractère nul "ISA(space)100(null)". L'identificateur CryptoKeyIdentifier pour cette clé est 0.
- Clé de rattachement (K_join): clé reçue à la conclusion de l'étape de configuration, qui est utilisée pour rejoindre un réseau pour lequel l'appareil est configuré. La valeur par défaut de la clé K_join est la même que la valeur par défaut de la clé K_global.
- Clé principale (master key): clé dérivée à la conclusion d'un plan d'agrément de clé, qui est utilisée comme KEK pour la communication entre le gestionnaire de sécurité et l'appareil, ainsi que comme base pour dériver d'autres clés. Cette clé expire et a besoin d'être mise à jour périodiquement.
- Clé D: clé utilisée pour chiffrer/déchiffrer et/ou authentifier des DPDU. Cette clé expire et a besoin d'être mise à jour périodiquement.
- Clé T: clé utilisée pour chiffrer/déchiffrer et/ou authentifier des TPDU. Cette clé expire et a besoin d'être mise à jour périodiquement.

7.2.2.3 Clés asymétriques et certificats

La prise en charge de la cryptographie asymétrique est une option de la construction d'un appareil.

Toutes les clés asymétriques WISN doivent avoir une force cryptographique d'au moins 128 bits. Les clés asymétriques utilisées comprennent:

- CA_root: La clé publique de l'autorité de certification qui a signé le certificat de clés asymétriques de l'appareil. Cette clé est communément appelée clé racine; elle est

utilisée lors de la vérification de la vraie identité de l'appareil communiquant le certificat, ainsi que d'un certain nombre d'informations de codage connexes.

- Cert-A: Le certificat de clés asymétriques de l'appareil A, utilisé pour apporter la preuve de la vraie identité de l'appareil, ainsi que d'informations de codage connexes. Elle est utilisée pendant l'exécution d'un protocole authentifié d'établissement de clé à clés asymétriques.

La description du support cryptographique à clés asymétriques est fournie à l'Article H.3.

7.2.2.4 Durée de vie d'une clé

7.2.2.4.1 Généralités

Les clés symétriques sont limitées par une durée de vie et il convient qu'elles soient invalidées après que la durée de vie expire. Pour maintenir la sécurité des communications en cours, les clés de DL et de TL courantes sont mises à jour à intervalles réguliers pendant l'exploitation du système. Les clés de DL sont partagées parmi toutes les DLE communicantes d'un sous-réseau DL tandis que les clés de TL sont partagées uniquement entre une paire (ou potentiellement un ensemble plus grand) de TLE ayant précédemment établi une relation de communications TL. A l'opposé, les KEK sont partagées uniquement entre un appareil et le gestionnaire de système.

Dans la présente spécification, les durées de vie des clés (et les informations connexes) sont définies comme suit:

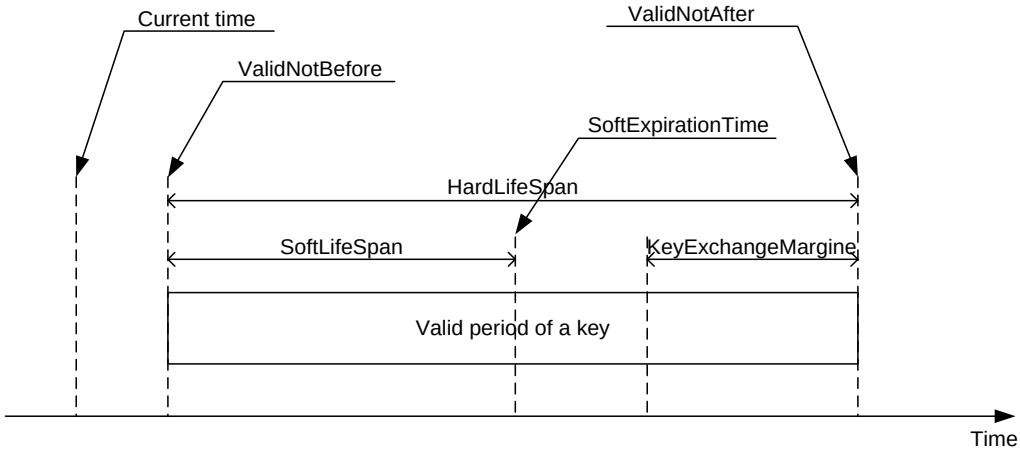
ValidNotBefore:	temps TAI absolu auquel une clé sera activée;
ValidNotAfter:	temps TAI après lequel une clé deviendra non valide;
SoftExpirationTime:	temps TAI auquel il convient qu'un appareil se prépare pour mettre à jour une clé;
HardLifeSpan:	durée relative de ValidNotBefore à ValidNotAfter;
KeyExchangeMargin:	temps minimal exigé pour achever un cycle de mise à jour de clé.

NOTE 1 Etant donné que les valeurs ci-dessus sont utilisées dans le présent document en tant que variables dans des formules, elles utilisent la typographie applicable aux variables.

La relation des définitions des durées de vie ci-dessus est illustrée à la Figure 36. Le mécanisme de mise à jour de clé utilisant ces définitions de temps est décrit en 7.6.

La valeur spéciale 0xFFFF FFFF est utilisée pour désigner des clés qui n'expirent jamais; elle est utilisée pour les clés globales spécifiées en 7.2.2.2. Ainsi, tout calcul du délai d'expiration d'une clé doit remplacer une valeur de résultat 0xFFFF FFFF par 0x0000 0000. De la même manière, toute logique qui détermine si une clé a expiré du fait que le délai d'expiration de la clé se situe dans un passé proche doit déterminer que l'expiration n'est pas arrivée lorsque cette valeur pour le délai d'expiration est 0xFFFF FFFF.

NOTE 2 Les clés de DL, de TL et les clés KEK (maître de l'appareil) sont plus sûres si elles expirent, car les clés qui n'expirent pas augmentent la vulnérabilité du système aux attaques et observations prolongées.



Légende

Anglais	Français
Current time	Instant courant
Valid period of a key	Période valide d'une clé
Time	Temps

Figure 36 – Durées de vie des clés

NOTE 3 Une clé utilisée après sa dure durée de vie peut rendre les communications vulnérables aux attaques par rejet.

Il convient que les certificats à clés asymétriques aient une durée de vie (ValidNotBefore et ValidNotAfter), conformément à la définition en 7.4.6.2.1.1.

La durée de vie KeyExchangeMargin peut être utilisée comme un déclencheur pour invoquer la méthode PSMO.Key_Update_Request() pour garder la session sécurisée continue. Il convient que la valeur de KeyExchangeMargin soit mise à "5 fois DSMO.pduMaxAge" secondes, soit:

- 2 × DSMO.pduMaxAge secondes pour une communication aller-retour par la méthode Security_New_Session();
- 2 × DSMO.pduMaxAge secondes pour une communication aller-retour par la méthode New_Key(); et
- d'autres DSMO.pduMaxAge secondes pour la durée de traitement.

7.2.2.4.2 Expiration de la durée de vie d'une clé

7.2.2.4.2.1 SoftExpirationTime

Lorsque SoftExpirationTime est passé, l'appareil possédant la clé se prépare pour obtenir une nouvelle clé issue du gestionnaire de sécurité. L'appareil peut appeler la méthode PSMO.Security_New_Session() sur le gestionnaire de système pour demander explicitement une clé, ou il peut attendre que sa méthode DSMO.New_Key() soit appelée par le gestionnaire de sécurité. Si l'appareil souhaite être sûr de la mise au jour d'une clé, il convient qu'il appelle explicitement la méthode PSMO.Security_New_Key() sur le gestionnaire de système.

Il n'est pas nécessaire que l'appareil commence le processus de mise à jour de clé immédiatement après l'expiration de SoftExpirationTime. La mise à jour de clé peut être accomplie à tout moment jusqu'à ValidNotAfter. Pour garder la session sécurisée courante avec un homologue, une demande de nouvelle clé peut être émise à tout moment entre SoftExpirationTime et ValidNotAfter. Il convient que l'appareil appelle la méthode PSMO.Security_New_Key() avant (HardExpirationTime – KeyExchangeMargin).

7.2.2.4.2.2 ValidNotAfter

La clé ne doit pas être utilisée dans la communication active après ValidNotAfter et il convient que cette clé soit abrogée par tous les appareils utilisant la clé. Cependant, la clé peut être archivée d'une façon sécurisée, selon la politique d'archivage de clés du système.

7.3 Sécurité de PDU

7.3.1 Généralités

7.3.1.1 Niveau de sécurité

Le niveau de sécurité spécifie la méthode devant être appliquée à certaines PDU. Le niveau de sécurité est constitué d'une combinaison de la taille du MIC (0 bit, 32 bits, 64 bits ou 128 bits) et la question de savoir si la charge utile de PDU associée doit être chiffrée ou non. Le Tableau 35 montre les niveaux de sécurité utilisés dans la présente spécification et leurs attributs de sécurité correspondants.

Tableau 35 – Niveaux de sécurité

Valeur de niveau de sécurité	Attributs de sécurité	Utilisation
0	Aucune	TPDU
1	MIC-32	DPDU Data, DPDU ACK/NAK, TPDU
2	MIC-64	DPDU Data, TPDU
3	MIC-128	TPDU
4	ENC-only	Jamais
5	ENC-MIC-32	TPDU, DPDU Data
6	ENC-MIC-64	TPDU, DPDU Data
7	ENC-MIC-128	TPDU

NOTE 1 Les contraintes de taille et les taux de perte des PhPDU dictent la restriction des DPDU ACK/NAK au mode MIC-32 et la restriction des DPDU Data aux modes MIC-32, ENC-MIC-32, MIC-64 et ENC-MIC-64.

NOTE 2 Les DPDU ACK/NAK ne comportent pas de champ de charge utile auquel le mode ENC pourrait s'appliquer.

NOTE 3 Le mode ENC-only est exclu, car on ne peut pas déterminer si le déchiffrement éventuel est correct.

7.3.1.2 Champ contrôle de sécurité

Le champ contrôle de sécurité fait partie de chaque en-tête de sécurité de DL et de TL. Sa valeur spécifie la présence de l'identificateur de clé et le niveau de sécurité devant être appliqué à la PDU. Les octets du champ SecurityControl doivent être conformes à l'IEEE 802.15.4:2011, 7.4.1.

Le Tableau 36 montre la structure du champ contrôle de sécurité.

Tableau 36 – Structure du champ contrôle de sécurité

Octet	Bits							
	7	6	5	4	3	2	1	0
1	Réservé			CryptoKeyIdentifierMode		Niveau de sécurité		

Le champ CryptoKeyIdentifierMode code la taille du champ CryptoKeyIdentifier qui suit immédiatement le champ SecurityControl dans la PDU. Si le mode d'identificateur de clé est mis à 0, le champ suivant CryptoKeyIdentifier est élidé.

Le champ niveau de sécurité doit être constitué de 3 bits tels que définis dans l'IEEE 802.15.4:2011, Tableau 58, et récapitulés dans le Tableau 35 de la présente norme. Le niveau de sécurité 0x04, correspondant au chiffrage seulement, ne doit jamais être utilisé pour une TPDU, ou la première DPDU d'une transaction D, de la présente norme. Le niveau de sécurité 0x00, correspondant à l'absence de protection, ne doit jamais être utilisé pour une DPDU dans la présente norme.

NOTE Le mode encryption-only (chiffrage seulement) ne fournit aucune protection contre un attaquant actif, car un tel attaquant est capable de complémenter arbitrairement des bits sélectionnés de n'importe quelle PDU en transit. Sans champ d'intégrité dur du point de vue cryptographique, il n'y a aucune méthode sécurisée pour que le destinataire détecte un tel changement et, donc, n'importe quel attaquant actif peut facilement fabriquer une PDU malveillante.

7.3.2 Sécurité de DPDU

7.3.2.1 Généralités

Le degré auquel un appareil est autorisé à participer à un sous-réseau D doit être déterminé par la politique du système appliquée aux authentifiants fournis par l'appareil. Les appareils sans authentifiants doivent être autorisés à une participation pleine, limitée ou nulle au-delà de tentatives de rattachement, telle que déterminée par la politique du système pour de tels appareils.

Toutes les DPDU incluent des champs de sécurité et un DMIC fort du point de vue cryptographique. Les détails des blocs modules de base cryptographiques sont montrés en Annexe H. En mode non sécurisé, la clé distribuée pourrait avoir voyagé sur une voie non sécurisée. Lorsqu'une clé D secrète correctement sécurisée est utilisée, les services de sécurité suivants sont toujours fournis:

a) authentification établie à la source de la DPDU;

b) intégrité de la DPDU; et

c) preuve que la DPDU était reçue à l'instant prévu, assurant le rejet des DPDU

   – dont la source n'était pas un appareil au sein du réseau qui partage une clé de données appropriée, ou

   – qui n'avaient pas été reçues dans la limite d'une fenêtre de temps acceptable par rapport à leur heure de formation ou d'émission, ou

   – qui avaient été reçues précédemment.

NOTE 1 L'autorisation est sous-entendue par le fait que l'appareil expéditeur a connaissance d'une clé de données symétrique partagée. Lorsque la clé n'est pas un secret partagé, l'autorisation s'étend à tous les appareils possibles par l'utilisation de la clé globale. Lorsque la clé est un secret partagé, une inférence est disponible que l'appareil expéditeur a obtenu la clé secrète partagée en provenance d'un gestionnaire de sécurité, et qu'il aurait obtenu la clé seulement si la base de données d'autorisations du gestionnaire de sécurité avait permis la relation de communication protégée qui en résulte. Une telle permission est habituellement basée sur le rôle de l'appareil. Voir Device_Role_Capability (identificateur de type d'objet normalisé 127, identificateur d'attribut 4, dans le Tableau 10) pour une définition des rôles et de leur carte de bits respective.

NOTE 2 La détection de la réception à un instant inadéquat rend inefficaces les attaques contre le flux de messages MAC qui sont basées sur le retard, le réarrangement ou le rejet de DPDU, car la durée d'émission de chaque DPDU est supérieure à la fenêtre de 1 ms dans la limite de laquelle un tel réarrangement aurait été indétectable.

NOTE 3 Ce service utilise l'heure d'émission de l'expéditeur, l'heure de réception du destinataire, et le fait que l'émission et la réception MAC sont fortement concomitantes pour assurer que toute DPDU reçue à un instant intempestif, y compris le rejet de DPDU ou le réarrangement de flux de DPD, sera détectée et la(les) DPDU anachronique(s) rejetée(s).

La quantité de redondance (à savoir la taille de DMIC) qui est utilisée pour fournir l'intégrité de DPDU est sélectionnée par la politique associée à la clé de données pertinente.

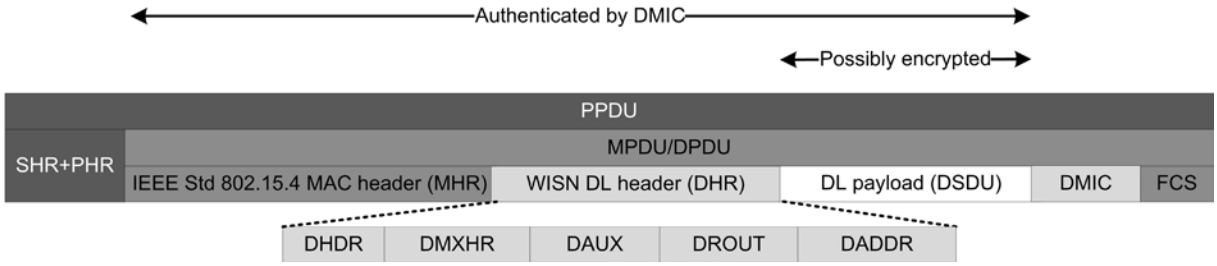
Le service de sécurité de DL complémentaire suivant est sélectionnable par la politique associée à la clé D pertinente:

d) Confidentialité de la charge utile des DPDU (à savoir le chiffrement).

Ce service de confidentialité ne doit pas être offert avec les clés K_open et K_global spécifiées en 7.2.2.2, car l'utilisation de ces clés avec leurs valeurs constantes bien connues rend la confidentialité impossible.

7.3.2.2 Structure de DPDU

La structure d'une DPDU est décrite en 9.3.1, et présentée dans les grandes lignes à la Figure 88 et à la Figure 37 dans la présente norme, avec la DSDU éventuellement chiffrée et les MHR, DHR et DSDU protégés par le DMIC.



Légende

Anglais	Français
Authenticated by DMIC	Authentifié par DMIC
Possibly encrypted	Possiblement chiffré
IEEE Std892.15.4 MAC header (MHR)	En-tête (MHR) de MAC selon l'IEEE std892.15.4
WISN DL header (DHR)	En-tête WISN DL (DHR)
DL payload (DSDU)	Charge utile de DL (DSDU)

Figure 37 – Structure de DPDU

La DPDU complète, du début du MHR jusqu'à la fin de la DSDU, doit être protégée par le DMIC. Les informations pertinentes pour le DSC sont l'en-tête d'extension MAC de la DL (DMXHR) tel que présenté dans le Tableau 112, le numéro de séquence de la DPDU de 8 bits tel que présenté dans le Tableau 110 et le numéro de voie de la PhPDU (dans la plage 0..15).

7.3.2.3 En-têtes de DPDU

7.3.2.3.1 En-tête MAC IEEE 802.15.4

La sécurité de la DPDU est fournie par la pile de protocoles définie dans la présente norme, au-dessus de la sous-couche MAC IEEE 802.15.4. L'en-tête MAC est défini en 9.3.3.2.

7.3.2.3.2 En-tête d'extension MAC de la DL

Le DMXHR présenté dans le Tableau 112 doit contenir 2 champs utilisés par la couche de sécurité. Le premier champ doit contenir le champ contrôle de sécurité tel que présenté dans les grandes lignes en 7.3.1.2. Le second champ doit contenir le CryptoKeyIdentifier tel que spécifié dans l'IEEE 802.15.4:2011, 7.4.3. Dans le DMXHR, le CryptoKeyIdentifier ne doit jamais être élidé avec le CryptoKeyIdentifierMode = 0.

La valeur par défaut du niveau de sécurité pour la DL doit être mise à 1 (MIC-32), correspondant à l'authentification uniquement avec une taille de DMIC de 32 bits.

Pour les étapes de traitement des DPDU, les contraintes suivantes doivent être respectées:

- Les tailles de DMIC de 0 bit et de 128 bits sont interdites, ce qui interdit donc les niveaux de sécurité de DPDU de 0 (none), 3 (MIC-128), 4 (ENC-only) et 7 (ENC-MIC-128).

NOTE 1 Le niveau MIC-64 garantit une protection adéquate pour les DPDU Data, étant donné leur petite taille maximale. Cette contrainte de taille rend la protection MIC-128 problématique tandis que le taux d'erreurs des PhPDU sous-jacentes dicte l'utilisation de MIC pour une meilleure intégrité des DPDU. Un MIC assure également une protection statistique contre du brouillage commis par une personne malveillante qui ne connaît pas la clé de chiffrement symétrique pertinente.

NOTE 2 Le niveau ENC-only n'est pas utile, car on ne peut pas déterminer à la réception si la DPDU est reçue intacte.

- Les DPDU ACK/NAK ne doivent utiliser que des DMIC de 32 bits, quel que soit le niveau de sécurité des DPDU Data d'une transaction D.

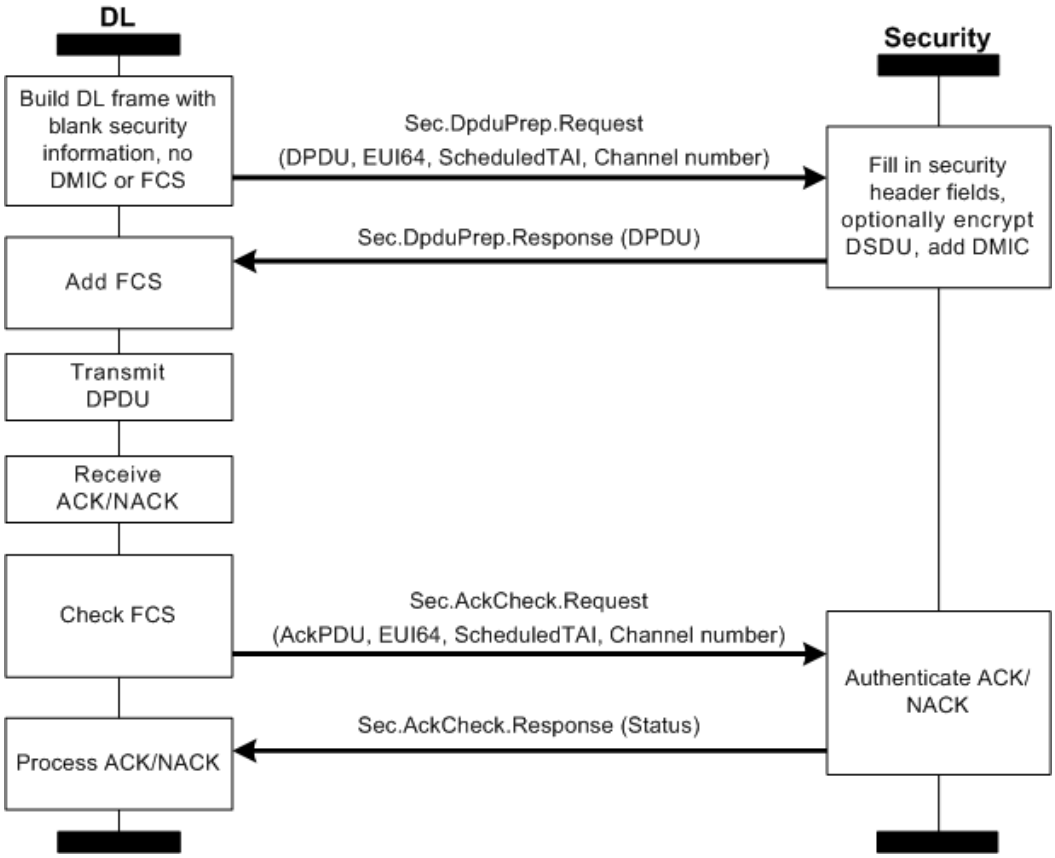
NOTE 3 Le niveau MIC-32 garantit une protection adéquate pour les DPDU Data, étant donné leur taille minimale et les contraintes réglementaires en ce qui concerne la durée des messages SCS auxquels ils s'appliquent. Les DPDU ACK/NAK ne transportent aucune charge utile à laquelle les fonctionnalités ENC (chiffrement) de la DL pourraient être appliquées.

7.3.2.4 Interface entre la DLE et le DSC

7.3.2.4.1 Généralités

Le Figure 38 résume la relation entre la DLE et le DSC pour les transactions de DPDU. Cet écoulement couvre le cas normal où une DPDU est émise et acquittée et aucune erreur ne se produit. Pour plus de détail, voir la documentation pour les DSAP correspondants en 7.3.2.4.2, 7.3.2.4.3, 7.3.2.4.4, 7.3.2.4.5, 7.3.2.4.6, 7.3.2.4.7, 7.3.2.4.8 et 7.3.2.4.9.

Toutes les interfaces entre la DLE et le DSC sont des interfaces internes au sein de la DLE, et sont donc inobservables. Par conséquent, elles ne sont pas soumises à une normalisation.



Légende

Anglais	Français
DL	DL
Build DL frame with blank security information, no DMIC or FCS	Bâtir une trame DL avec des informations de sécurité vides, pas de DMIC ou FCS

Anglais	Français
Add FCS	Ajouter FCS
Transmit DPDU	Emettre DPDU
Receive ACK/NACK	Recevoir ACK/NACK
Check FCS	Vérifier FCS
Process ACK/NACK	Traiter ACK/NACK
Security	Sécurité
Fill in security header fields, optionally encrypt DSDU, add DMIC	Remplir les champs d'en-tête de sécurité, facultativement chiffrer DSDU, ajouter DMIC
Authenticate ACK/NACK	Authentifier ACK/NACK

Figure 38 – DLE et traitement de DLS pour un initiateur de transaction D

Le DLE assemble les DPDU devant être protégées. Par convention de documentation, les champs de sécurité dans l'en-tête de la DPDU sont peuplés par le DSC.

Certaines informations de sécurité de DPDU sont fournies par la DLE au DSC:

- le temps TAI programmé de l'intervalle de temps, utilisé dans le nonce pour détecter les DPDU différées et reproduites;

NOTE 1 Le temps TAI programmé transmis au DSC est le temps de début TAI programmé de l'intervalle de temps auquel la transaction D est assignée. Le DSC tronque ce temps à 2^{-10} s (approximativement 1 ms).

NOTE 2 Il existe un scénario selon la présente norme où un seul appareil pourrait initier plusieurs émissions qui ont toutes le même temps de début d'intervalle de temps programmé. Dans ce cas, un appareil (habituellement un routeur dorsal) fonctionne simultanément sur plusieurs voies, en utilisant des modèles d'intervalle de temps synchronisé de telle manière qu'il puisse utiliser soit une seule antenne partagée, soit plusieurs antennes étroitement espacées, programmées de sorte que les émissions sur une ou plusieurs voies ne perturbent pas la réception sur d'autres voies. Alors qu'un tel fonctionnement n'est pas explicitement décrit par la norme, il est également intentionnellement non interdit. La prise en charge d'un tel fonctionnement donne naissance aux deux composants de nonce suivants qui sont inclus dans la construction du nonce de la DPDU.

- le numéro de voie de chaque DPDU, utilisé dans le nonce pour détecter les DPDU construites pour être utilisées dans une voie donnée qui sont rejouées dans le même intervalle de temps dans une autre voie;

NOTE 3 Le numéro de voie utilise la convention de numérotation de voies selon la présente norme, les numéros 0..15 correspondant respectivement aux voies 11..26 de l'IEEE 802.15.4.

- le numéro de séquence d'un seul octet trouvé dans l'en-tête MAC, utilisé dans le nonce pour différencier entre les DPDU Data d'une transaction D et les éventuelles DPDU ACK/NAK qui pourraient être générées dans des intervalles de temps avec le même temps de début TAI programmé;

NOTE 4 Les bits de poids faible de l'octet du numéro de séquence de MHR codent la position d'origine zéro de la DPDU dans la transaction D: 0 pour la DPDU Data, 1 pour la première DPDU ACK/NAK, 2 pour la deuxième DPDU ACK/NAK, etc.

- le DSC a besoin de l'adresse EUI64Address de l'appareil de destination afin de traiter sa DPDU ACK/NAK;

NOTE 5 Lorsqu'elle est connue de la DLE, cette adresse D est récupérée directement de la table dlmo.Neighbor.

- lorsque l'adresse EUI64Address d'une destination en monodiffusion n'est pas connue de la DLE, l'indicateur EUI64Address-requested (adresse EUI64Address demandée) dans l'octet de contrôle de trame de DHDR (Tableau 111) doit être mis à 1), ce qui incite la destination à retourner son adresse EUI64Address dans la DPDU ACK/NAK.

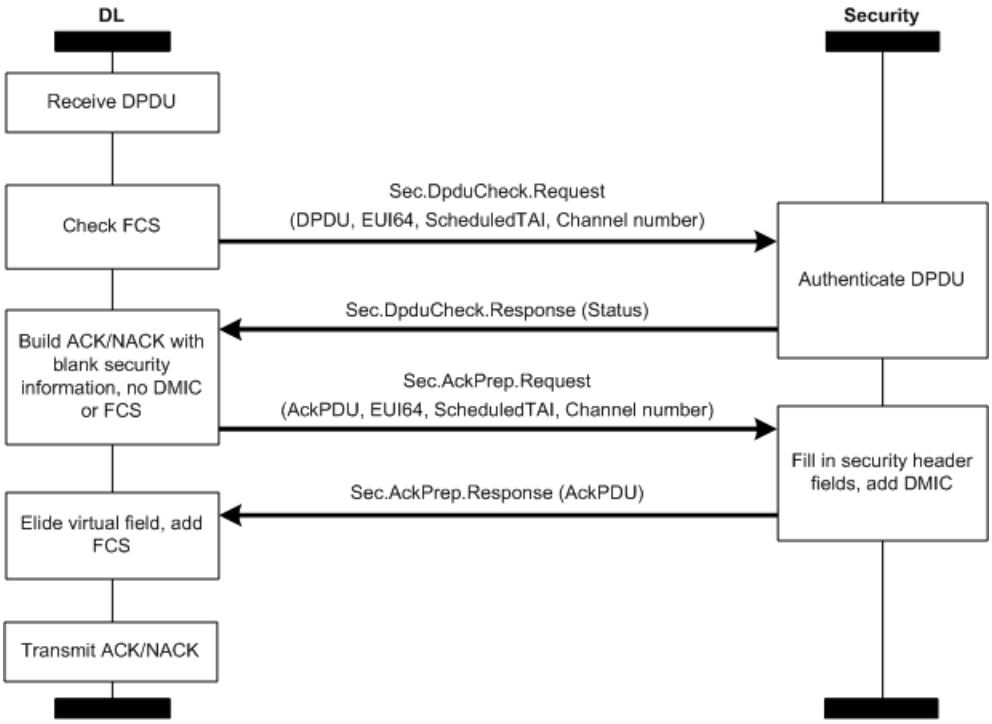
NOTE 6 Le DSC utilise la taille de DSDU pour chiffrer seulement la DSDU et pas l'en-tête de la DPDU, alors que le DMIC protège la DPDU entière. Ce détail n'est pas montré à la Figure 38 ou Figure 39.

La DLE maintient une copie du DMIC sortant à des fins d'utilisation ultérieure pour relier sans ambiguïté les DPDU ACK/NAK de réponse aux DPDU Data de la transaction D. La DLE aboute alors une FCS IEEE 802.15.4 à la DPDU et l'émet sans retard indu.

Lorsque la DLE reçoit une DPDU ACK/NAK, elle demande au DSC d'authentifier la DPDU. Certaines informations de sécurité de DPDU sont fournies par la DLE au DSC:

- Chaque DPDU ACK/NAK doit faire écho au DMIC de la DPDU initiale de la transaction D comme un champ virtuel (voir Tableau 117) dans le calcul de son MIC D. La DPDU ACK/NAK complète, incluant ce champ virtuel, est reconstruite par la DLE avant qu'elle ne soit vérifiée par le DSC.
- L'adresse EUI64Address de l'émetteur de la DPDU ACK/NAK est soit obtenue par consultation, soit fournie au sein de la DPDU ACK/NAK elle-même.
- Le temps TAI programmé du début de l'intervalle de temps de la transaction D, qui est habituellement le même que le temps de début d'intervalle de temps TAI utilisé par l'initiateur de transaction D. Cependant, lorsque le saut de voie lent est utilisé, la DPDU ACK/NAK peut inclure un décalage d'intervalle de temps (voir 9.3.4), auquel cas le nonce formé pour vérifier la DPDU ACK/NAK doit utiliser le temps de début TAI programmé de l'intervalle de temps référencé par le décalage d'intervalle de temps; à savoir, l'intervalle de temps programmé de la DLE donnant acquittement.
- Le numéro de voie pour envoyer la DPDU ACK/NAK est fourni au DSC.
- Le numéro de séquence de MHR est fourni au DSC de la même manière qu'il est fourni pour les DPDU Data de la transaction D.

La Figure 39 illustre la relation entre une DLE et son DSC pour les transactions D dans lesquelles la DLE est un destinataire ou un répondeur des DPDU Data de la transaction D.



Légende

Anglais	Français
Receive DPDU	Recevoir DPDU
Check FCS	Vérifier FCS
Build ACK/NACK with blank security information, no DMIC or FCS	Bâtir ACK/NACK avec des informations de sécurité vides, pas de DMIC ni de FCS

Anglais	Français
Elide virtual field, add FCS	Elider le champ virtuel, ajouter FCS
Transmit ACK/NACK	Emettre ACK/NAK
Security	Sécurité
Authenticate DPDU	Authentifier la DPDU
Fill in security header fields, add DMIC	Remplir les champs d'en-tête de sécurité, ajouter DMIC

Figure 39 – DPDU reçues – DLE et DSC

Lorsqu'elle reçoit une DPDU, la DLE envoie une demande à la DSE d'authentifier la DPDU et, s'il y a lieu, de déchiffrer la charge utile de la DPDU. Le temps TAI programmé et le numéro de voie sont inclus avec cette demande. L'adresse EUI64Address de la source de la DPDU a besoin d'être connue a priori par la DLE, à l'exception du cas d'une demande de rattachement où elle est transportée dans l'en-tête de la DPDU comme adresse source. Le DSC répond normalement avec une authentification positive.

La DLE construit la DPDU ACK/NAK. La DPDU ACK/NAK doit utiliser le même temps TAI programmé que les DPDU Data reçues de la transaction D, excepté lorsque la correction de décalage de lent saut de voie est fournie dans la DPDU ACK/NAK comme débattu ci-dessus. La DPDU ACK/NAK doit également faire écho au DMIC de la DPDU initialement reçue de la transaction D comme un champ virtuel, conformément au Tableau 117. Le DSC sécurise alors la DPDU ACK/NAK, incluant le DMIC de la DPDU comme un champ virtuel. La DLE élide le champ virtuel, aboute une FCS IEEE 802.15.4 et émet la DPDU ACK/NAK.

Lorsque le sens local du temps d'une DLE est corrigé par une DPDU ACK, faisant que son temps est réinitialisé à un intervalle de temps antérieur, il doit y avoir une pause forcée du service, égale à l'ampleur de la correction d'intervalle de temps plus au moins un intervalle de temps.

7.3.2.4.2 Sec.DpduPrep.Request

7.3.2.4.2.1 Généralités

Sec.DpduPrep.Request ordonne au DSC de protéger une unité de données de protocole de DL selon le cas.

7.3.2.4.2.2 Sémantique de la primitive de service

La sémantique de Sec.DpduPrep.Request est comme suit:

```
Sec.DpduPrep.Request    (
    DPDU,
    EUI64,
    ScheduledTAI,
    ChannelNumber,
    AckHandle)
```

Le Tableau 37 spécifie les éléments pour la Sec.DpduPrep.Request.

Tableau 37 – Eléments de la Sec.DpduPrep.Request

Nom de l'élément	Identificateur de l'élément	Type de l'élément scalaire
DPDU (la DPDU devant être émise)	1	Type: OctetString
EUI64 (adresse EUI64Address de l'appareil expéditeur)	2	Type: EUI64Address
ScheduledTAI (32 bits du temps de début de l'intervalle de temps tronqué à une résolution de 2^{-10} s)	3	Type: Unsigned32
ChannelNumber (le numéro de voie utilisée dans la DPDU émise)	4	Type: Unsigned8 Plage valide: 0..15
AckHandle (abstraction qui connecte chaque invocation de Sec.DpduPrep.Request avec le rappel subséquent par Sec.DpduPrep.Response)	5	Type: Extrait

Le DSC fournit à la DLE le contrôle de sécurité (octet 1) approprié et le CryptoKeyIdentifier (octet 2) obtenu à partir du KeyDescriptor pour la clé D courante, pour être utilisés dans le sous-en-tête du DMXHR de la DPDU, dont le format est décrit en 9.3.3.4. Voir 7.3.2.5 sur la sélection de la clé D correcte.

Le DSC peuple le champ DMIC comme spécifié par la politique de la clé D sélectionnée.

7.3.2.4.2.3 Utilisation appropriée

La DLE invoque la primitive Sec.DpduPrep.Request pour ajouter de la protection de sécurité à une DPDU avant qu'elle ne soit émise.

7.3.2.4.2.4 Effet à la réception

A la réception de la primitive Sec.DpduPrep.Request, le DSC commence les étapes appropriées de traitement des DPDU pour protéger la DPDU conformément à ce que dicte la politique.

7.3.2.4.3 Sec.DpduPrep.Response

7.3.2.4.3.1 Généralités

Sec.DpduPrep.Response rend compte du résultat d'une Sec.DpduPrep.Request.

7.3.2.4.3.2 Sémantique

La sémantique de Sec.DpduPrep.Response est comme suit:

```
Sec.DpduPrep.Response    (
    DPDU,
    Status,
    AckHandle)
```

Le Tableau 38 spécifie les éléments pour Sec.DpduPrep.Response.

Tableau 38 – Eléments de Sec.DpduPrep.Response

Nom de l'élément	Identificateur de l'élément	Type de l'élément scalaire
DPDU	1	Type: OctetString
Status (le résultat d'une primitive Sec.DpduPrep.Request)	2	Type: Unsigned Valeurs nommées: 0: réussite; > 0: échec
AckHandle (abstraction qui connecte chaque invocation de Sec.DpduPrep.Request avec le rappel subséquent par Sec.DpduPrep.Response)	3	Type: Extrait

7.3.2.4.3.3 Moment de sa génération

Le DSC génère la Sec.DpduPrep.Response en réponse à une Sec.DpduPrep.Request. La Sec.DpduPrep.Response retourne une valeur de statut qui indique soit SUCCESS et la DPDU peu sécurisée, soit le code d'erreur approprié.

7.3.2.4.3.4 Utilisation appropriée

A la réception de Sec.DpduPrep.Response, la DL reçoit notification du résultat de la demande de protéger une DPDU sortante.

7.3.2.4.4 Sec.DAckCheck.Request

7.3.2.4.4.1 Généralités

Sec.DAckCheck.Request ordonne au DSC de protéger une DPDU ACK/NAK entrante.

7.3.2.4.4.2 Sémantique de la primitive de service

La sémantique de Sec.DAckCheck.Request est comme suit:

```
Sec.DAckCheck.Request     (
    AckPDU,
    EUI64,
    ScheduledTAI,
    ChannelNumber,
    AckHandle)
```

Le Tableau 39 spécifie les éléments pour la Sec.DAckCheck.Request.

Tableau 39 – Eléments de Sec.DAckCheck.Request

Nom de l'élément	Identificateur de l'élément	Type de l'élément scalaire
AckPDU (l'AckPDU devant être vérifiée)	1	Type: OctetString
EUI64 (adresse EUI64Address de l'appareil donnant acquittement)	2	Type: EUI64Address
ScheduledTAI (32 bits du temps de début de l'intervalle de temps tronqué à une résolution de 2^{-10} s)	3	Type: Unsigned32
ChannelNumber (le numéro de voie utilisé pour recevoir la DPDU ACK/NAK entrante)	4	Type: Unsigned Plage valide: 0..15
AckHandle (abstraction qui connecte chaque invocation de Sec.DAckCheck.Request avec le rappel subséquent par Sec.DAckCheck.Response)	5	Type: Extrait

Le DSC vérifie que le DHR de la PDU ACK/NAK DPDU a utilisé le mode DMIC (voir Tableau 118) spécifié par la politique de clé D courante. La clé D utilisée pour authentifier la DPDU ACK/NAK est la même que celle utilisée pour les DPDU Data de la transaction D.

Le DSC vérifie le champ DMIC comme dicté par les étapes de traitement des DPDU et les politiques courantes.

7.3.2.4.4.3 Utilisation appropriée

La DLE invoque la primitive Sec.DAckCheck.Request pour vérifier une DPDU ACK/NAK après sa réception.

7.3.2.4.4.4 Effet à la réception

A la réception de la primitive Sec.DAckCheck.Request, le DSC exécute les étapes appropriées du traitement des DPDU tel que spécifié en 7.3.2.6 pour vérifier la DPDU ACK/NAK reçue.

7.3.2.4.5 Sec.DAckCheck.Response

7.3.2.4.5.1 Généralités

Sec.DAckCheck.Response rend compte du résultat d'une Sec.DAckCheck.Request.

7.3.2.4.5.2 Sémantique

La sémantique de Sec.DAckCheck.Response est comme suit:

```
Sec.DInitialCheck.Response (
    AckPDU,
    Status,
    AckHandle)
```

Le Tableau 40 spécifie les éléments pour Sec.DAckCheck.Response.

Tableau 40 – Eléments de Sec.DAckCheck.Response

Nom de l'élément	Identificateur de l'élément	Type de l'élément scalaire
AckPDU	1	Type: OctetString
Status (le résultat d'une primitive Sec.DAckPrep.Request)	2	Type: Unsigned Valeurs nommées: 0: réussite; > 0: échec
AckHandle (abstraction qui connecte chaque invocation de Sec.DAckCheck.Request avec le rappel subséquent par Sec.DAckCheck.Response)	3	Type: Extrait

7.3.2.4.5.3 Moment de sa génération

Le DSC génère la Sec.DAckCheck.Response en réponse à une Sec.DAckCheck.Request. La Sec.DAckCheck.Response retourne une valeur de statut qui indique soit SUCCESS, soit le code d'erreur approprié.

7.3.2.4.5.4 Utilisation appropriée

A la réception de Sec.DAckCheck.Response, la DL reçoit notification du résultat de la vérification et du déchiffrement éventuel d'une DPDU entrante.

7.3.2.4.6 Sec.DInitialCheck.Request

7.3.2.4.6.1 Généralités

Sec.DInitialCheck.Request ordonne au DSC de vérifier et éventuellement de déchiffrer une unité de données de protocole de DL entrante selon le cas.

7.3.2.4.6.2 Sémantique de la primitive de service

La sémantique de Sec.DInitialCheck.Request est comme suit:

```
Sec.DInitialCheck.Request   (
    DPDU,
    EUI64,
    ScheduledTAI,
    ChannelNumber,
    AckHandle)
```

Le Tableau 41 spécifie les éléments pour la Sec.DInitialCheck.Request.

Tableau 41 – Eléments de Sec.DInitialCheck.Request

Nom de l'élément	Identificateur de l'élément	Type de l'élément scalaire
DPDU (la DPDU devant être vérifiée et éventuellement déchiffrée)	1	Type: OctetString
EUI64 (adresse EUI64Address de l'appareil expéditeur)	2	Type: EUI64Address
ScheduledTAI (32 bits du temps de début de l'intervalle de temps tronqué à une résolution de 2^{-10} s)	3	Type: Unsigned32
ChannelNumber (le numéro de voie utilisé pour recevoir la DPDU entrante)	4	Type: Unsigned Plage valide: 0..15
AckHandle (abstraction qui connecte chaque invocation de Sec.DInitialCheck.Request avec le rappel subséquent par Sec.DInitialCheck.Response)	5	Type: Extrait

Le DSC vérifie que le DMXHR de la DPDU a le contrôle de sécurité (octet 1) approprié en le comparant à la politique courante. Le CryptoKeyIdentifier (octet 2) est utilisé pour récupérer le support de clé correct. Voir 7.3.2.6.

Le DSC vérifie le champ DMIC comme dicté par les politiques courantes.

7.3.2.4.6.3 Utilisation appropriée

La DL invoque la primitive Sec.DInitialCheck.Request pour vérifier et éventuellement déchiffrer une DPDU avant qu'elle ne soit émise.

7.3.2.4.6.4 Effet à la réception

A la réception de la primitive Sec.DInitialCheck.Request, le DSC commence les étapes appropriées de traitement des PDU pour vérifier la DPDU entrante comme cela est dicté par les étapes de traitement des PDU entrantes en 7.3.2.6.

7.3.2.4.7 Sec.DInitialCheck.Response

7.3.2.4.7.1 Généralités

Sec.DInitialCheck.Response rend compte du résultat d'une Sec.DInitialCheck.Request.

7.3.2.4.7.2 Sémantique

La sémantique de Sec.DInitialCheck.Response est comme suit:

```
Sec.DInitialCheck.Response (
                    DPDU,
                    Status,
                    AckHandle)
```

Le Tableau 42 spécifie les éléments pour Sec.DInitialCheck.Response.

Tableau 42 – Eléments de Sec.DInitialCheck.Response

Nom de l'élément	Identificateur de l'élément	Type de l'élément scalaire
DPDU	1	Type: OctetString
Status (le résultat d'une primitive Sec.DpduPrep.Request)	2	Type: Unsigned Valeurs nommées: 0: réussite; > 0: échec
AckHandle (abstraction qui connecte chaque invocation de Sec.DInitialCheck.Request avec le rappel subséquent par Sec.DInitialCheck.Response)	3	Type: Extrait

7.3.2.4.7.3 Moment de sa génération

Le DSC génère la Sec.DInitialCheck.Response en réponse à une Sec.DInitialCheck.Request. La Sec.DInitialCheck.Response retourne une valeur de statut qui indique soit SUCCESS, soit le code d'erreur approprié.

7.3.2.4.7.4 Utilisation appropriée

A la réception de Sec.DInitialCheck.Response, la DL reçoit notification du résultat de la vérification et du déchiffrement éventuel d'une DPDU entrante.

7.3.2.4.8 Sec.DAckPrep.Request

7.3.2.4.8.1 Généralités

Sec.DAckPrep.Request ordonne au DSC de protéger une DPDU ACK/NAK selon le cas.

7.3.2.4.8.2 Sémantique de la primitive de service

La sémantique de Sec.DAckPrep.Request est comme suit:

```
Sec.DAckPrep.Request      (
    AckPDU,
    EUI64,
    ScheduledTAI,
    ChannelNumber,
    AckHandle)
```

Le Tableau 43 spécifie les éléments pour la Sec.DAckPrep.Request.

Tableau 43 – Eléments de Sec.DAckPrep.Request

Nom de l'élément	Identificateur de l'élément	Type de l'élément scalaire
AckPDU (inclut l'en-tête virtuel)	1	Type: OctetString
EUI64 (adresse EUI64Address de l'appareil donnant acquittement)	2	Type: EUI64Address
ScheduledTAI (32 bits du temps de début de l'intervalle de temps tronqué à une résolution de 2^{-10} s)	3	Type: Unsigned32
ChannelNumber (le numéro de voie utilisé pour émettre la DPDU ACK/NAK)	4	Type: Unsigned8 Plage valide: 0..15
AckHandle (abstraction qui connecte chaque invocation de Sec.DAckPrep.Request avec le rappel subséquent par Sec.DAckPrep.Response)	5	Type: Extrait

Le DSC peuple la DPDU ACK/NAK avec le contrôle de sécurité (octet 1) approprié tel que décrit dans le Tableau 118. Dans le cas où plusieurs clés D sont actuellement valides, la clé utilisée pour authentifier la DPDU ACK/NAK est la même qui est utilisée pour la DPDU correspondante pour cette DPDU ACK/NAK.

Le DSC peuple le champ DMIC comme dicté par les politiques courantes. Noter que le champ DMIC dans une PDU ACK/NAK est toujours de 32 bits.

7.3.2.4.8.3 Utilisation appropriée

La DL invoque la primitive Sec.DAckPrep.Request pour protéger une DPDU ACK/NAK avant qu'elle ne soit émise.

7.3.2.4.8.4 Effet à la réception

A la réception de la primitive Sec.DAckPrep.Request, le DSC commence les étapes appropriées de traitement des PDU pour protéger la DPDU ACK/NAK conformément à ce que dicte la politique. Noter que la DPDU ACK/NAK est seulement authentifiée et n'est jamais chiffrée.

7.3.2.4.9 Sec.DAckPrep.Response

7.3.2.4.9.1 Généralités

Sec.DAckPrep.Response rend compte du résultat d'une Sec.DAckPrep.Request.

7.3.2.4.9.2 Sémantique

La sémantique de Sec.DAckPrep.Response est comme suit:

```
Sec.DAckPrep.Response    (
    AckPDU,
    Status,
    AckHandle)
```

Le Tableau 44 spécifie les éléments pour Sec.DAckPrep.Response.

Tableau 44 – Eléments de Sec.DAckPrep.Response

Nom de l'élément	Identificateur de l'élément	Type de l'élément scalaire
AckPDU	1	Type: OctetString
Status (le résultat d'une primitive Sec.DAckPrep.Request)	2	Type: Unsigned Valeurs nommées: 0: réussite; > 0: échec
AckHandle (abstraction qui connecte chaque invocation de Sec.DAckPrep.Request avec le rappel subséquent par Sec.DAckPrep.Response)	3	Type: Extrait

7.3.2.4.9.3 Moment de sa génération

Le DSC génère la Sec.DAckPrep.Response en réponse à une Sec.DAckPrep.Request. La Sec.DAckPrep.Response retourne une valeur de statut qui indique soit SUCCESS, soit le code d'erreur approprié.

7.3.2.4.9.4 Utilisation appropriée

A la réception de Sec.DAckPrep.Response, la DL reçoit notification du résultat de la demande de vérifier une AckPDU entrante.

7.3.2.4.10 Construction du nonce pour les DPDU

La présente norme utilise une construction de nonce de DPDU différente de celle de l'IEEE 802.15.4. Un nonce de 13 octets est exigé pour le moteur CCM*. Le nonce doit être construit comme une concaténation allant du premier (le plus à gauche) octet au dernier (le plus à droite) octet des champs de données conformément au Tableau 45, où:

- l'adresse EUI64Address doit être utilisée comme une matrice de 8 octets (dans la convention MSB) de la même manière que l'adresse source du nonce CCM* dans l'IEEE 802.15.4:2011, 7.3.2;
- le temps TAI doit être représenté par les 32 bits de poids faible du temps TAI en unités de 2^{-10} s suivant la description dans le Tableau 46;
- le dernier octet doit être construit comme suit:
  - Le bit 7 doit être zéro, réservant ainsi la valeur 0xFF pour la couche transport (voir Tableau 57).
  - Les bits 6..3 (4 bits) doivent indiquer la voie radio d'émission, dans la plage 0..15, correspondant aux numéros de voies 11..26 de l'IEEE 802.15.4, dans le même ordre.
  - Les bits 2..0 doivent être copiés à partir des 3 bits de poids faible correspondants du numéro de séquence du MHR.

Tableau 45 – Structure du nonce de DPDU WISN

<table>
<tr><th rowspan="2">Octet</th><th colspan="8">Bits</th></tr>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>1</td><td colspan="8" rowspan="3">Adresse EUI64Address de l'initiateur de la DPDU</td></tr>
<tr><td>…</td></tr>
<tr><td>8</td></tr>
<tr><td>9</td><td colspan="8" rowspan="3">32 bits de poids faible du temps TAI du début d'intervalle de temps nominal (en unités de 2^{-10} s)</td></tr>
<tr><td>…</td></tr>
<tr><td>12</td></tr>
<tr><td>13</td><td>Réservé = 0</td><td colspan="3">Numéro de voie (0..15)</td><td colspan="4">3 bits de poids faible du numéro de séquence du MHR</td></tr>
</table>

Le temps TAI utilisé doit être une représentation fractionnaire à point fixe tronquée sur 32 bits du temps TAI à une granularité de 2^{-10} s et un intervalle de mesure de 2^{22} s. Avec cette représentation, il se passera plus de 48,5 jours avant que la même valeur du temps TAI ne se reproduise. Donc, la durée de vie maximale d'une clé D doit être de 48,5 jours avant qu'il ne soit nécessaire de déployer une nouvelle clé D. Le temps TAI pour cette opération doit être celui maintenu par la DLE.

NOTE 1 Il est important que la valeur de la représentation 32 bits du temps TAI ne se reproduise pas pendant la durée de vie d'une clé symétrique secrète pertinente afin d'éviter une collision potentielle de nonce avec la réutilisation qui entraînerait un keystream identique.

La représentation dans le nonce D de ce temps TAI de 32 bits tronqué, spécifié à 2^{-10} s, est décrite dans le Tableau 46.

Tableau 46 – Structure du temps TAI tronqué de 32 bits utilisé dans le nonce D-

Octet	Bits							
	7	6	5	4	3	2	1	0
1	Temps TAI tronqué (bits avec un poids de $2^{21}..2^{14}$ s)							
2	Temps TAI tronqué (bits avec un poids de $2^{13}..2^{6}$ s)							
3	Temps TAI tronqué (bits avec un poids de $2^{5}..2^{-2}$ s)							
4	Temps TAI tronqué (bits avec un poids de $2^{-3}..2^{-10}$ s)							

Les 3 bits de poids faible du numéro de séquence du MHR, ainsi que le numéro de voie, sont utilisés pour construire le dernier octet d'un nonce D. La DLE expéditrice doit assurer que les bits du numéro de séquence du MHR utilisés dans le nonce D sont uniques parmi tous ceux qu'il génère dans la limite du même intervalle de 2^{-10} s pour la même voie et la même clé D (voir 9.3.3.2 et 9.3.4). La valeur de 0xFF ne doit pas être utilisée pour le MHR. Sachant que ce nonce D a tout au plus huit valeurs distinctes pour une voie donnée et un intervalle de 2^{-10} s, une DLE ne doit pas émettre plus de huit DPDU par 2^{-10} s sur la même voie en utilisant la même clé D.

NOTE 2 L'inclusion du numéro de voie dans le nonce D fournit la prise en charge pour les appareils qui fonctionnent simultanément sur plusieurs voies.

NOTE 3 La construction du numéro de séquence du MHR est décrite en 9.3.3.2.

7.3.2.5 Traitement d'une DPDU devant être émise

Les entrées à la procédure de sécurité des DPDU sont:

- la DPDU devant être sécurisée;
- l'adresse EUI64Address de la DLE source;
- le temps de début TAI nominal de l'intervalle de temps utilisé pour la transaction D;
- l'octet du numéro de séquence du MHR; et
- le numéro de voie (0..15) devant être utilisé pour la transaction D.

Les sorties délivrées par cette procédure sont:

- le statut de la procédure; et
- si ce statut est succès, la DPDU sécurisée.

La procédure de sécurité pour les DPDU qui sont construites pour l'émission est constituée des étapes suivantes:

a) La procédure doit obtenir le KeyDescriptor issu du Tableau 93 satisfaisant aux critères de sélection suivants:

   1) Les entrées avec KeyUsage = '0x00' (autrement dit la clé D).Dans le cas initial, où une DLE en rattachement n'a aucun KeyDescriptor, l'appareil en rattachement crée un KeyDescriptor avec K_global. Le KeyDescriptor doit inclure au moins les paramètres suivants:

      CryptoKeyIdentifier = 0

      Security Level = 0x01 (MIC-32)

      KeyUsage = 0x00 (clé de groupe pour traitement de PDU)

      Key lifetime = never-expires (0xFFFF FFFF)

   2) De ces entrées, les entrées valides pour la période courante, satisfaisant à l'inégalité

      ValidNotBefore < temps courant < ValidNotAfter

      doivent être sélectionnées. Si aucune n'est disponible, la procédure doit retourner avec un statut UNAVAILABLE_KEY.

3) De ces entrées, si au moins deux clés sont valides pour le temps courant et la procédure avait été appelée à partir d'une DAckPrep.Request ou DAckCheck.Request, la procédure doit sélectionner la clé utilisée pour authentifier les DPDU Data de la transaction D.

   Autrement, si au moins deux clés sont valides pour le temps courant, la procédure doit sélectionner la clé avec la plus grande valeur de ValidNotAfter.

4) De ces entrées, si au moins deux clés ont le même ValidNotAfter, la procédure doit sélectionner la clé avec le plus grand ValidNotBefore.

5) De ces entrées, si au moins deux clés ont le même SoftExpirationTime, la procédure doit sélectionner la clé avec le CryptoKeyIdentifier le plus élevé.

b) La procédure doit récupérer la politique dans le KeyDescriptor sélectionné.

c) La procédure doit déterminer si la DPDU devant être sécurisée satisfait à la contrainte relative à la taille maximale des DPDU, comme suit:

1) La procédure doit établir la taille M, en octets, du champ d'authentification de DMIC à partir du niveau de sécurité.

2) Le champ CryptoKeyIdentifierMode dans le DMXHR doit avoir la valeur 1. Si le DMXHR inclut le champ décalage d'intervalle de temps de saut lent de voie, la taille de DMXHR est de 3 octets; autrement, elle est de 2 octets.

3) La procédure doit déterminer l'expansion de données qui en résulte comme étant (DMXHR_size + M).

4) La procédure doit vérifier si la taille de la DPDU devant être sécurisée, y compris l'expansion des données, est inférieure ou égale à la taille de DPDU maximale. Si cette vérification échoue, la procédure doit retourner un statut de DPDU_TOO_LONG.

d) La procédure doit utiliser le temps TAI programmé du début de l'intervalle de temps, tel que décrit dans le Tableau 46. S'il y a un potentiel pour que l'appareil envoie plusieurs DPDU avec la même valeur de temps TAI, alors la procédure doit sélectionner une valeur devant être acheminée dans l'en-tête du MHR de la DPDU qui est différente de toutes les autres valeurs de ce type lancées par l'appareil à cette valeur particulière du temps TAI. Une procédure pour déterminer le numéro de séquence à partir duquel le MHR est dérivé est définie en 9.3.3.2.

e) La procédure doit insérer le DMXHR dans la DPDU tel que décrit dans les grandes lignes dans le Tableau 112, avec les champs mis comme suit:

1) Le sous-champ niveau de sécurité du champ contrôle de sécurité doit être mis au niveau de sécurité 001 par défaut.

2) Le sous-champ CryptoKeyIdentifierMode du champ contrôle de sécurité doit être mis au paramètre CryptoKeyIdentifierMode 01 par défaut.

f) La procédure doit mettre l'octet CryptoKeyIdentifier dans le DMXHR. Voir Tableau 112.

g) La procédure doit insérer le numéro de séquence du MHR dans le MHR de DPDU Data. Voir Tableau 110.

h) La procédure doit utiliser l'adresse EUI64Address de l'appareil émetteur, les 32 bits de poids faible du temps TAI en 2^{-10} s, les 3 bits de poids faible du numéro de séquence du MHR, et le numéro de voie pour construire le nonce selon le Tableau 45.

i) La procédure doit utiliser le nonce, le support de clé, l'en-tête, la charge utile et le mode CCM* de fonctionnement tels que décrits dans l'IEEE 802.15.4:2011, 7.3.4, pour sécuriser la DPDU:

1) Si le paramètre SecurityLevel spécifie l'utilisation du chiffrement (voir IEEE 802.15.4:2011, Tableau 58), l'opération de chiffrement doit être appliquée seulement au champ de la charge utile de la DPDU. Le champ de charge utile correspondant est passé au processus de transformation CCM* décrit dans l'IEEE 802.15.4:2011, 7.3.4, comme étant la charge utile non sécurisée. La charge utile chiffrée résultante doit être substituée à la charge utile d'origine.

2) Les champs restants dans la DPDU, jusqu'au champ de charge utile non compris, doivent être passés au processus de transformation CCM* décrit dans l'IEEE 802.15.4:2011, 7.3.4, comme étant le champ non-charge utile.

3) L'ordonnancement et la manière exacte d'accomplir les opérations relatives au chiffrement et à l'intégrité ainsi que le placement des données chiffrées ou du code d'intégrité ainsi obtenus au sein du champ de charge utile de la DPDU doivent être tels que définis dans l'IEEE 802.15.4:2011, 7.3.4.

j) La procédure doit retourner la DPDU sécurisée et un statut de SUCCESS.

7.3.2.6 Traitement des DPDU reçues

Les entrées à la procédure de sécurité pour les DPDU reçues sont la DPDU devant être dépourvue de sécurité, le numéro de voie sur lequel la DPDU a été reçue, et le temps TAI nominal du début de l'intervalle de temps dans lequel la DPDU a été reçue. Les sorties délivrées par cette procédure sont la DPDU non sécurisée, le niveau de sécurité, le CryptoKeyIdentifierMode, le CryptoKeyIdentifier et le statut de la procédure. Toutes les sorties de cette procédure sont supposées être non valides, jusqu'à ce que cela soit explicitement établi dans cette procédure. Il est supposé que les KeyDescriptors avec un seul et unique appareil ou un certain nombre d'appareils ont été établis par le DSMO.

La procédure de sécurité à la réception des DPDU est constituée des étapes suivantes:

a) La procédure doit mettre le niveau de sécurité et le CryptoKeyIdentifierMode aux valeurs des sous-champs correspondants du champ contrôle de sécurité du DMXHR de la DPDU entrante, et le CryptoKeyIdentifier aux valeurs des sous-champs correspondants du champ CryptoKeyIdentifier du DMXHR de la DPDU devant être dépourvue de sécurité.

b) La procédure doit obtenir le KeyDescriptor issu du Tableau 93 satisfaisant aux critères de sélection suivants:

1) Les entrées avec KeyUsage = '0x00' (autrement dit la clé D). Dans le cas initial, où un appareil en rattachement n'a aucun KeyDescriptor, l'appareil en rattachement crée un KeyDescriptor temporaire avec K_global. Le KeyDescriptor doit inclure au moins les paramètres suivants:

CryptoKeyIdentifier = 0

Security Level = 0x01 (MIC-32)

KeyUsage = 0x00 (clé de groupe pour traitement de PDU)

Key lifetime = never-expires (0xFFFF FFFF)

NOTE L'utilisation du KeyDescriptor pour K_global est décrite en 9.1.10.

2) De ces entrées, l'entrée avec le CryptoKeyIdentifier concordant au CryptoKeyIdentifier de la PDU entrante doit être sélectionnée.

3) Si cette procédure échoue, la procédure doit retourner avec un statut UNAVAILABLE_KEY.

c) La procédure doit déterminer si le niveau de sécurité de la DPDU entrante se conforme à la politique de niveau de sécurité en comparant le SecurityLevel du KeyDescriptor concordant obtenu dans l'étape b) ci-dessus. S'il y a une discordance, la procédure doit retourner avec un statut IMPROPER_SECURITY_LEVEL.

d) Si la durée de vie dans le KeyDescriptor est finie (> 0x0000), la procédure doit vérifier que le numéro de séquence du MHR de 8 bits n'a pas été reçu précédemment pour la même valeur de l'adresse EUI64Address de la source, la même représentation fractionnaire à point fixe sur 32 bits du temps TAI et la même clé. Si cette vérification échoue, la procédure doit retourner avec un statut DUPLICATE_DPDU.

e) La procédure doit alors utiliser l'adresse EUI64Address de l'expéditeur, le temps TAI programmé, et les 3 bits de poids faible du numéro de séquence du MHR, et le numéro de voie pour générer le nonce conformément au Tableau 45. En plus, la procédure doit vérifier que le numéro de séquence du MHR de 8 bits n'est pas 0xFF. Si le numéro de

séquence du MHR de 8 bits est 0xFF, la procédure doit retourner avec un statut INVALID_SEQUENCE_NUMBER.

f) La procédure doit utiliser le nonce, la crypto clé issue du KeyDescriptor obtenu dans l'étape b), les en-têtes réels (les champs non-charge utile), la charge utile et le MIC de la DPDU entrante et le mode de fonctionnement CCM* tels que décrits dans les opérations (voir l'IEEE 802.15.4:2011, 7.3.5) pour authentifier et, si spécifié, déchiffrer la DPDU:

   1) Si le niveau de sécurité spécifie l'utilisation du chiffrement (voir l'IEEE 802.15.4:2011, Tableau 58), l'opération de déchiffrement doit être appliquée seulement au champ charge utile réel de la DPDU (voir l'IEEE 802.15.4:2011, 5.2.2.2.2). Le champ de charge utile correspondant doit être passé au processus de transformation inverse CCM* décrit dans l'IEEE 802.15.4:2011, 7.3.5, comme étant la charge utile sécurisée.

   2) Les champs restants dans la DPDU doivent être passés au processus de transformation inverse CCM* décrit dans l'IEEE 802.15.4:2011, 7.3.5, comme étant les champs non-charge utile (voir IEEE 802.15.4:2011, Tableau 57).

   3) L'ordonnancement et la manière exacte d'accomplir les opérations relatives au déchiffrement et à la vérification d'intégrité ainsi que le placement des données déchiffrées ainsi obtenues au sein du champ de charge utile de la DPDU doivent être tels que définis dans l'IEEE 802.15.4:2011, 7.3.5.

g) Si le processus de transformation inverse CCM* échoue, la procédure doit établir la DPDU non sécurisée comme étant la DPDU devant être dépourvue de sécurité et retourner un statut SECURITY_ERROR.

h) Si la durée de vie dans le KeyDescriptor expire, la procédure doit insérer la valeur nonce (inclut le numéro de séquence de MHR, le numéro de voie, l'adresse EUI64Address de la source et le temps TAI programmé) dans le champ NonceCache du KeyDescriptor correspondant, pour activer la protection contre le rejet.

i) La procédure doit retourner avec la DPDU non sécurisée, le niveau de sécurité, le CryptoKeyIdentifierMode, le CryptoKeyIdentifier et un statut SUCCESS.

7.3.2.7 Détection et rejet des unités de données de protocole dupliquées ou rejouées

Voir 7.3.2.6, d).

7.3.3 Fonctionnalité de sécurité de TL

7.3.3.1 Généralités

L'interaction du DSC et de la TL est décrite dans les grandes lignes. Les étapes de traitement de TL ont été écrites pour réutiliser les aspects communs entre la DL et la TL. Cependant, sachant que la DL et la TL existent à des couches d'abstraction du réseau différentes avec des exigences et des hypothèses différentes, il existe des différences considérables entre les étapes de traitement de la DL et de la TL.

Les services de sécurité à la TL sont sélectionnés par la politique associée à la clé pertinente de données de transport, obtenue comme partie intégrante d'une nouvelle demande de session ou d'une mise à jour de clé et basée sur la politique de transport maintenue par le gestionnaire de sécurité associée à tout ou partie des éléments suivants:

- l'appareil expéditeur;
- l'UAP demandeur; et/ou
- l'association de transport, telle que définie par ses points d'extrémité.

Le service de sécurité de transport suivant doit toujours être fourni avec une clé active:

- Communication autorisée avec authentification de TPDU, intégrité, et acheminement du temps nominal de création de TPDU, fournissant le rejet des TPDU périmées:
  – dont la source n'était pas un appareil au sein du réseau qui partage une clé de données appropriée; ou

– qui étaient gravement périmées, c'est-à-dire qui n'étaient pas reçues à un instant dans le délai maximal de DSMO.pduMaxAge secondes du temps nominal de création de la TPDU.

- Confidentialité de la charge utile de couche d'application au sein de la TPDU.

NOTE 1 Seize bits d'informations relatives au temps sont émis par chaque TPDU.

NOTE 2 Ces services utilisent le temps nominal d'émission de la source tel qu'authentifié au récepteur pour entraîner le rejet des TPDU qui sont excessivement retardées et pour fournir la détection des TPDU en doublons dans la limite de cette fenêtre de temps.

Le service de confidentialité ne doit pas être utilisé avec les clés qui ne sont pas des secrets partagés, car cela rendrait impossible la véritable confidentialité, et parce que cet aspect de la politique associée à de telles clés est constant.

Il convient que le MIC soit validé dans les limites de la période DSMO.pduMaxAge. Si la vérification échoue, la validation du MIC peut être répétée en décrémentant une fenêtre de temps pour récupérer le temps de création de la PDU.

7.3.3.2 Structure de TPDU

7.3.3.2.1 Généralités

La structure d'une TPDU est décrite en 11.5, et dans les grandes lignes à la Figure 109 et à la Figure 40 dans la présente norme, avec la TSDU éventuellement chiffrée et le contenu de l'en-tête UDP, de l'en-tête de sécurité et la TSDU protégés par le TMIC.

Légende

Anglais	Français
Authenticated by TMIC	Authentifié par TMIC
Possibly encrypted/decrypted	Possiblement chiffré/déchiffré
Uncompressed UDP header	En-tête UDP décompressé
Security header	En-tête de sécurité
Application Payload	Charge utile d'application

Figure 40 – Structure de TPDU et couverture protégée

La TPDU complète, du début de l'en-tête UDP jusqu'à la fin de l'Application Payload (charge utile d'application), doit être protégée par le TMIC. Chaque paramètre dans l'en-tête UDP est protégé en utilisant le pseudo-en-tête de composant de sécurité de transport (TSC) pour le calcul du TMIC. Le pseudo-en-tête de TSC est décrit en 7.3.3.2.2.

NOTE Le TSC est décrit en 11.2. Voir également la brève discussion des pseudo-en-têtes en 4.5.2.1.

7.3.3.2.2 Protection de TPDU

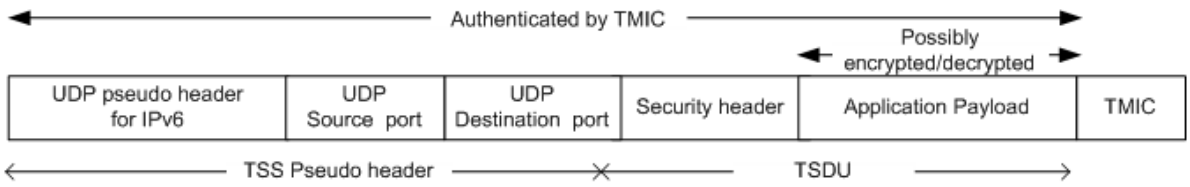
NOTE 1 L'utilisation d'IPV6 n'implique et ne fournit pas de connectivité Internet. Néanmoins, elle facilite l'utilisation d'outils de gestion IPvv6 communs au sein du réseau WISN.

Le TMIC est utilisé pour protéger l'information dans l'en-tête UDP, l'en-tête de sécurité de TL et la TSDU. Il protège également les adresses IPv6 source et de destination de la NL en utilisant une forme étendue du pseudo en-tête UDP pour IPv6. Le pseudo en-tête UDP pour IPv6 est décrit en 11.4.2 et dans le RFC IETF 2460:1998, 8.1. La taille de charge utile UDP et

la somme de contrôle (virtuelle) dans l'en-tête UDP ne sont pas utilisées pour le calcul du TMIC.

NOTE 2 La somme de contrôle et la taille de charge utile UDP n'apparaissent pas dans le pseudo-en-tête, car la somme de contrôle est élidée (absente de la TPDU) lorsque le TMIC est présent, et la taille de charge utile UDP est déterminée à partir de la longueur de la NSDU dans le pseudo-en-tête UDP pour IPv6.

Les paramètres pour le TMIC sont montrés à la Figure 41.



Légende

Anglais	Français
Authenticated by TMIC	Authentifié par TMIC
Possibly encrypted/decrypted	Possiblement chiffré/déchiffré
UDP pseudo-header for Ipv6	Pseudo-en-tête UDP pour IPv6
UDP source port	Port de source UDP
UDP destination port	Port de destination UDP
Security header	En-tête de sécurité
Application Payload	Charge utile d'application
TSS pseudo-header	Pseudo-en-tête TSS

Figure 41 – Paramètres de TMIC

Le TSC construit les paramètres de TMIC conformément à la Figure 41, avec la TPDU reçue, le temps TAI nominal auquel la TPDU a été créée, le KeyDescriptor et les informations de contrat fournies par la TL. Le TSC peut alors utiliser les paramètres pour l'opération de sécurité appropriée sur la TPDU.

L'en-tête IPv6 et les ports source et destination UDP sont passés de la TL au TSC. La combinaison de ces paramètres est un pseudo en-tête UDP étendu qui est appelé pseudo-en-tête de TSC dans la présente norme. La structure du pseudo-en-tête de TSC est montrée dans le Tableau 47. L'utilisation approprie est décrite en 7.3.3.5.4, en 7.3.3.5.5 et en 7.3.3.5.8.

Tableau 47 – Structure du pseudo-en-tête de TSC

Nom de l'élément	Identificateur de l'élément	Type de l'élément scalaire
Source IPv6Address	1	Type: IPv6Address Description: Adresse IPv6 décompressée de l'initiateur de la TPDU
Destination IPv6Address	2	Type: IPv6Address Description: Adresse IPv6 décompressée du destinataire prévu de la TPDU
NSDU size	3	Type: Unsigned16 Description: Longueur de NDSU en octets
Réservé	4	Type: Unsigned8 Description: Champ réservé. Actuellement rempli de 0
Next header	5	Type: Unsigned8 Plage valide: 17 (UDP) Description: Prochaine valeur d'en-tête dans l'en-tête IPv6. Il convient que cette valeur soit seulement de 17
UDP source port	6	Type: Unsigned16 Description: Numéro de port UDP source de l'initiateur de la TPDU
UDP destination port	7	Type: Unsigned16 Description: Numéro de port UDP de destination du destinataire prévu de la TPDU

7.3.3.3 Interface avec la TL pour une TPDU formée en vue de l'émission

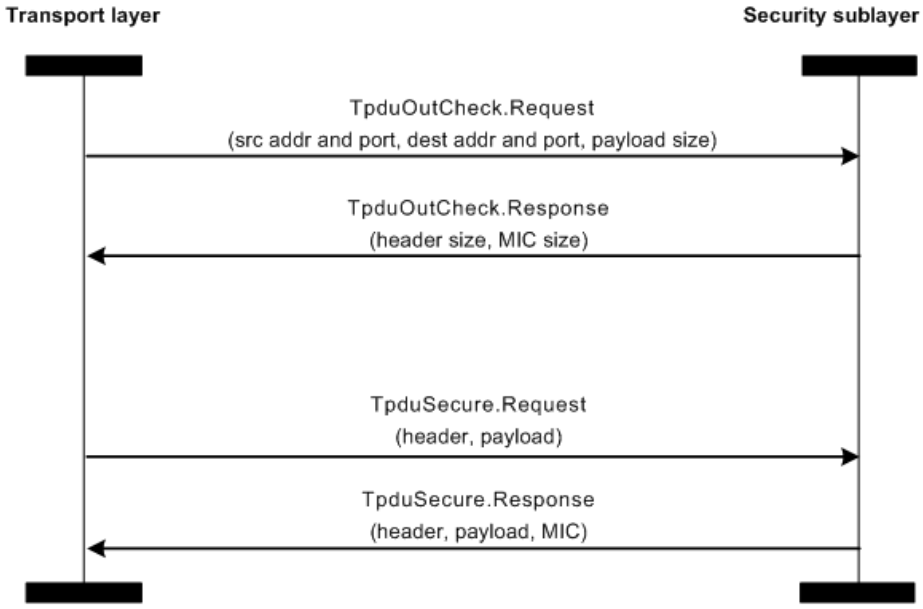
L'interaction de TL avec la couche de sécurité pour une TPDU formée en vue de l'émission est récapitulée à la Figure 42. Lorsque le TSC reçoit l'adresse de source, le port de source, l'adresse de destination, le port de destination, et la taille de charge utile, il exécute une consultation de la table de KeyDescriptor pour voir si, oui ou non, la sécurité est activée pour cette session particulière.

Si le niveau de la sécurité de la session est égal à 0: none, une taille d'en-tête de 3 (octets) et une taille de TMIC de 0 octet doivent être retournées.

NOTE Lorsque le niveau de sécurité est zéro, la somme de contrôle UDP normalisée et trivialement falsifiée est utilisée pour détecter les erreurs qui se produisent pendant l'acheminement de la TPDU.

Autrement le TSC doit retourner les tailles appropriées du CryptoKeyIdentifier et du TMIC. Toutes les sessions à la TL sont en monodiffusion; par conséquent, la taille du CryptoKeyIdentifier doit être 0 ou 1 en fonction du nombre de clés valides disponibles à cet instant pour cette association de sécurité.

La TL appelle alors le TSC avec l'en-tête et la charge utile. Selon la politique de sécurité pour cette session particulière, la charge utile peut être chiffrée, et un TMIC peut être généré. L'en-tête et la charge utile résultants seront retournés à la TL pour émission.



Légende

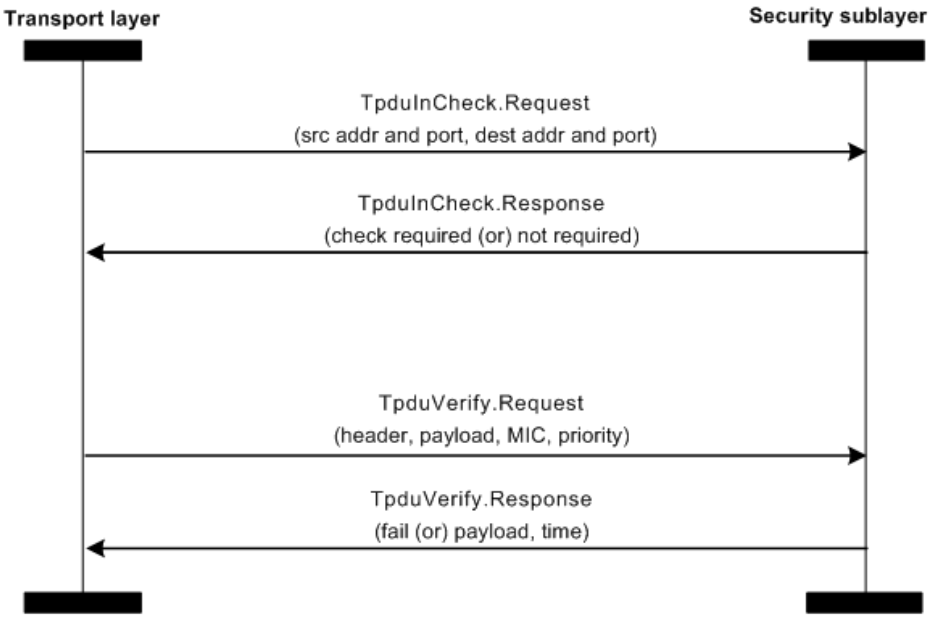
Anglais	Français
Transport layer	Couche transport
Security sub-layer	Sous-couche de sécurité
TpduOutCheck.Request (src addr and port, dest addr and port, payload size)	TpduOutCheck.Request (adr et port src, adr et port dest, charge utile)
TpduOutCheck.Response (header size, MIC size)	TpduOutCheck.Response (taille d'en-tête, taille de MIC)
TpduSecure.Request (header, payload)	TpduSecure.Request (en-tête, charge utile)
TpduSecure.Response (header, payload, MIC)	TpduSecure.Response (en-tête, charge utile, MIC)

Figure 42 – Interaction de la TL et du TSC, TPDU sortante

7.3.3.4 Vue d'ensemble du traitement pour les TPDU reçues

L'interaction de la TL avec la couche de sécurité pour une TPDU reçue est récapitulée à la Figure 43. Lorsque le TSC reçoit l'adresse source, le port source, l'adresse de destination, et le port de destination, il accomplit une consultation dans la MIB pour voir si la sécurité est activée pour cette session particulière et retourne SEC_CHECK_REQUIRED ou SEC_CHECK_NOT_REQUIRED.

La TL doit alors appeler le TSC avec l'en-tête, la charge utile, et le MIC. Selon la politique de sécurité pour cette session particulière, la charge utile peut être déchiffrée, et un MIC peut être vérifié. Si le contrôle de sécurité échoue, un statut FAILURE sera retourné, avec la charge utile retournée intacte. Si l'opération réussit, la charge utile résultante et le codage récupéré de la durée de TPDU seront retournés à la TL.



Légende

Anglais	Français
Transport layer	Couche transport
TpduInCheck.Request (src addr and port, dest addr and port)	TpduInCheck.Request (addr et port src, addr et port de dest)
TpduInCheck.Response (check required (or) not required)	TpduInCheck.Response (contrôle exigé (ou) pas exigé)
TpduVerify.Request (header, payload, MIC, priority)	TpduVerify.Request (en-tête, charge utile, MIC, priorité)
TpduVerufy.Response (fail (or) payload, time)	TpduVerufy.Response (échec (ou) charge utile, temps)
Security sub-layer	Sous-couche de sécurité

Figure 43 – Interaction de la TL et du TSC, TPDU entrante

7.3.3.5 Interface de la TL au TSC

7.3.3.5.1 Généralités

La relation entre la TL et le TSC est décrite dans les grandes lignes en 7.3.3.2 pour l'interface de la TL pour une TPDU sortante et en 7.3.3.4 pour l'interface de la TL pour une TPDU entrante.

7.3.3.5.2 Sec.TpduOutCheck.Request

7.3.3.5.2.1 Généralités

Sec.TpduOutCheck.Request est une vérification de la TL au TSC pour obtenir la taille des champs de sécurité (éventuels) exigé dans la TPDU sortante.

7.3.3.5.2.2 Sémantique de la primitive de service

La sémantique de Sec.TpduOutCheck.Request est comme suit:

```
Sec.TpduOutCheck.Request (
    Source_Address,
    Source_Port,
```

Destination_Address,

Destination_Port,

Payload_Size)

Le Tableau 48 spécifie les éléments pour la Sec.TpduOutCheck.Request.

Tableau 48 – Eléments de Sec.TpduOutCheck.Request

Nom de l'élément	Identificateur de l'élément	Type de l'élément scalaire
Source_Address	1	Type: IPv6Address Plage valide: toutes les valeurs avec réglage sur bit de poids fort
Source_Port	2	Type: Nombre entier
Destination_Address	3	Type: IPv6Address
Destination_Port	4	Type: Nombre entier
Payload_Size	5	Type: Nombre entier Plage valide: 0..Assigned_Max_TSDU_Size; voir 11.4.3.3

Le TSC doit utiliser Source_Address, Source_Port, Destination_Address, et Destination_Port pour récupérer la politique (éventuelle) appropriée pour l'association de sécurité.

7.3.3.5.2.3 Utilisation appropriée

La TL invoque la primitive Sec.TpduOutCheck.Request pour protéger une TPDU avant qu'elle ne soit émise.

7.3.3.5.2.4 Effet à la réception

A la réception de la primitive de Sec.TpduOutCheck.Request, le TSC détermine si la TPDU a besoin d'être protégée et renvoie les tailles correspondantes d'en-tête et de TMIC.

7.3.3.5.3 Sec.TpduOutCheck.Response

7.3.3.5.3.1 Généralités

Sec.TpduOutCheck.Response rend compte du résultat d'une Sec.TpduOutCheck.Request.

7.3.3.5.3.2 Sémantique

La sémantique de Sec.TpduOutCheck.Response est comme suit:

Sec. TpduOutCheck.Response (

Sec_Header_Size,

TMIC_Size)

Le Tableau 49 spécifie les éléments pour la Sec.TpduOutCheck.Response.

Tableau 49 – Eléments de Sec.TpduOutCheck.Response

Nom de l'élément	Identificateur de l'élément	Type de l'élément scalaire
Sec_Header_Size (la taille d'en-tête supplémentaire exigée par le TSC, en octets complets)	1	Type: Nombre entier Plage valide: 0..Assigned_Max_TSDU_Size; voir Tableau 30
TMIC_Size (la taille du Transport Integrity Code, en octets complets)	2	Type: Nombre entier Valeurs valides: 0, 4, 8 ou 16

7.3.3.5.3.3 Moment de sa génération

Le TSC génère la Sec.TpduOutCheck.Response en réponse à une Sec.TpduOutCheck.Request. La Sec.TpduOutCheck.Response retourne les tailles complémentaires exigées pour prendre en charge la fonctionnalité de couche de sécurité. Une association de sécurité est une association de TL sécurisée basée sur:

- l'adresse source;
- le port source;
- l'adresse de destination;
- le port destination.

7.3.3.5.3.4 Utilisation appropriée

A la réception de Sec.TpduOutCheck.Response, la TL reçoit notification de la nécessité d'appliquer une opération de sécurité sur la TPDU, avec les octets complémentaires exigés pour prendre en charge cette opération.

7.3.3.5.4 Sec.TpduSecure.Request

7.3.3.5.4.1 Généralités

Sec.TpduSecure.Request ordonne au TSC d'exécuter les étapes appropriées pour sécuriser une TPDU sortante. Les informations relatives à l'association de sécurité sont contenues dans le pseudo-en-tête passé au TSC. Voir la Figure 108 en 11.4.2.

7.3.3.5.4.2 Sémantique de la primitive de service

La sémantique de Sec.TpduSecure.Request est comme suit:

```
Sec. TpduSecure.Request    (
    TSC_Pseudo_Header,
    TSC_Pseudo_Header_Size,
    TSDU,
    TSDU_Size)
```

Le Tableau 50 spécifie les éléments pour la Sec.TpduSecure.Request.

Tableau 50 – Eléments de Sec.TpduSecure.Request

Nom de l'élément	Identificateur de l'élément	Type de l'élément scalaire
TSC_Pseudo_Header	1	Type: OctetStringN
TSC_Pseudo_Header_Size	2	Type: Nombre entier Plage valide: 0..127
TSDU	3	Type: OctetStringN
TSDU_Size	4	Type: Nombre entier Plage valide: 0..Assigned_Max_TSDU_Size; voir Tableau 30

Le TSC doit obtenir l'adresse source, le port source, l'adresse de destination, et le port de destination issus du pseudo-en-tête de TSC. Ces informations sont utilisées pour récupérer le support et les politiques de codage appropriées pour cette association de sécurité.

Le TSC inclut les informations relatives à la priorité dans l'en-tête de TPDU, protège la confidentialité des données de la TPDU et génère le champ TMIC comme dicté par les étapes de traitement des PDU et les politiques courantes.

Le TSC peuple l'en-tête de sécurité de TL tel que spécifié dans les étapes de traitement des TPDU et les politiques.

7.3.3.5.4.3 Utilisation appropriée

La TL appelle la primitive de Sec.TpduSecure.Request pour protéger une TPDU sortante après que la TL a reçu le nombre d'octets complémentaires exigés dans l'en-tête de transport et le TMIC.

7.3.3.5.4.4 Effet à la réception

A la réception de la primitive Sec.TpduSecure.Request, le TSC commence les étapes appropriées de traitement des PDU pour protéger la TPDU sortante comme cela est dicté par les étapes de traitement des TPDU sortantes en 7.3.3.8.

7.3.3.5.5 Sec.TpduSecure.Response

7.3.3.5.5.1 Généralités

Sec.TpduSecure.Response rend compte du résultat d'une Sec.TpduSecure.Request.

7.3.3.5.5.2 Sémantique

La sémantique de Sec.TpduSecure.Response est comme suit:

```
Sec. TpduSecure.Response (
    TSC_Pseudo_Header,
    TSC_Pseudo_Header_Size,
    TSDU,
    TSDU_Size,
    TMIC,
    TMIC_Size,
    Status)
```

Le Tableau 51 spécifie les éléments pour Sec.TpduSecure.Response.

Tableau 51 – Eléments Sec. TpduSecure.Response

Nom de l'élément	Identificateur de l'élément	Type de l'élément scalaire
TSC_Pseudo_Header	1	Type: OctetStringN
TSC_Pseudo_Header_Size	2	Type: Unsigned7 Plage valide: 0..127
TSDU	3	Type: OctetStringN
TSDU_Size	4	Type: Unsigned Plage valide: 0..Assigned_Max_TSDU_Size; voir Tableau 30
TMIC	5	Type: OctetStringN
TMIC_Size	6	Type: Nombre entier Valeurs valides: 0, 4, 8, 16
Status	7	Type: Unsigned Valeurs nommées: 0: réussite; > 0: échec

7.3.3.5.5.3 Moment de sa génération

Le TSC génère la Sec.TpduSecure.Response en réponse à une Sec.TpduSecure.Request. La Sec.TpduSecure.Response retourne l'en-tête peuplé de transport de sécurité, une TSDU possiblement chiffrée et un TMIC avec les tailles appropriées. Enfin la Sec.TpduSecure.Response retourne une valeur de statut qui indique soit SUCCESS, soit le code d'erreur approprié.

7.3.3.5.5.4 Utilisation appropriée

A la réception de Sec.TpduSecure.Response, la TL reçoit notification du résultat de la protection d'une TPDU sortante.

7.3.3.5.6 Sec.TpduInCheck.Request

7.3.3.5.6.1 Généralités

Sec.TpduInCheck.Request ordonne au TDSC de vérifier et éventuellement de déchiffrer une unité de données de protocole de TL entrante selon le cas.

7.3.3.5.6.2 Sémantique de la primitive de service

La sémantique de Sec. TpduInCheck.Request est comme suit:

```
Sec.TpduInCheck.Request   (
    Source_Address,
    Source_Port,
    Destination_Address,
    Destination_Port,
    Payload_Size)
```

Le Tableau 52 spécifie les éléments pour la Sec.TpduInCheck.Request.

Tableau 52 – Eléments de Sec.TpduInCheck.Request

Nom de l'élément	Identificateur de l'élément	Type de l'élément scalaire
Source_Address	1	Type: IPv6Address
Source_Port	2	Type: Unsigned16
Destination_Address	3	Type: IPv6Address
Destination_Port	4	Type: Unsigned16
Payload_Size	5	Type: Unsigned Plage valide: 0..Assigned_Max_TSDU_Size

Le TSC utilise Source_Address, Source_Port, Destination_Address, et Destination_Port pour récupérer la politique (éventuelle) appropriée pour l'association de sécurité.

7.3.3.5.6.3 Utilisation appropriée

La TL invoque la primitive Sec.TpduInCheck.Request pour vérifier si une vérification sécurisée et possiblement un déchiffrement d'une TPDU à l'avance sont exigés.

7.3.3.5.6.4 Effet à la réception

A la réception de la primitive de Sec.TpduInCheck.Request, le TSC détermine si la TPDU a besoin d'être vérifiée et, potentiellement, déchiffrée et retourne un statut de succès ou d'échec.

7.3.3.5.7 Sec. TpduInCheck.Response

7.3.3.5.7.1 Généralités

Sec.TpduInCheck.Response rend compte du résultat d'une Sec.TpduInCheck.Request.

7.3.3.5.7.2 Sémantique

La sémantique de Sec.TpduInCheck.Response est comme suit:

```
Sec.TpduInCheck.Response (
    Status)
```

Le Tableau 53 spécifie les éléments pour Sec.TpduInCheck.Response.

Tableau 53 – Eléments de Sec.TpduInCheck.Response

Nom de l'élément	Identificateur de l'élément	Type de l'élément scalaire
Status (le résultat d'une primitive Sec.TpduInCheck.Request)	1	Type: Unsigned Valeurs nommées: 0: réussite; > 0: échec

7.3.3.5.7.3 Moment de sa génération

Le TSC génère la Sec.TpduInCheck.Response en réponse à une Sec.TpduInCheck.Request. La Sec.TpduInCheck.Response retourne une valeur de statut qui indique soit TRUE, soit FALSE, en fonction des politiques sur l'association de sécurité courante.

7.3.3.5.7.4 Utilisation appropriée

A la réception de Sec.TpduInCheck.Response, la TL reçoit notification de la nécessité d'appeler la Sec.TpduVerify.Request pour vérifier et possiblement déchiffrer la TPDU entrante.

7.3.3.5.8 Sec.TpduVerify.Request

7.3.3.5.8.1 Généralités

Sec.TpduVerify.Request ordonne au TSC de vérifier et, si configuré à cet effet, déchiffrer une TPDU entrante.

7.3.3.5.8.2 Sémantique de la primitive de service

La sémantique de Sec.TpduVerify.Request est comme suit:

```
Sec.TpduVerify.Request      (
    TSC_Pseudo_Header,
    TSC_Pseudo_Header_Size,
    TSDU,
    TSDU_Size,
    TMIC,
    TMIC_Size)
```

Le Tableau 54 spécifie les éléments pour la Sec.TpduVerify.Request.

Tableau 54 – Eléments de Sec.TpduVerify.Request

Nom de l'élément	Identificateur de l'élément	Type de l'élément scalaire
TSC_Pseudo_Header	1	Type: OctetString
TSC_Pseudo_Header_Size	2	Type: Unsigned Plage valide: 0..127
TSDU	3	Type: OctetString
TSDU_Size	4	Type: Unsigned Plage valide: 0..Assigned_Max_TSDU_Size; voir Tableau 30
TMIC	5	Type: OctetString
TMIC_Size	6	Type: Nombre entier Valeurs valides: 0, 4, 8 ou 16
Priority	7	Type: Unsigned4

Le TSC vérifie que le TL Security Header (en-tête de sécurité de TL) de la TPDU a le contrôle de sécurité (octet 1) approprié en le comparant à la politique courante. Le CryptoKeyIdentifier (octet 2), s'il est présent, est utilisé pour récupérer le support de clé correct. Voir 7.3.3.9.

Le TSC vérifie le champ TMIC comme dicté par les politiques courantes.

Le temps acheminé dans l'en-tête de sécurité de TL est utilisé dans la construction de nonce pour l'authentification et, si configuré, le déchiffrement de la TPDU reçue.

La priorité est fournie au TSC afin de permettre une mise en œuvre efficace de la protection contre le rejet.

7.3.3.5.8.3 Utilisation appropriée

La TL invoque la primitive Sec.TpduVerify.Request pour vérifier et éventuellement déchiffrer une TPDU avant qu'elle ne soit émise.

7.3.3.5.8.4 Effet à la réception

A la réception de la primitive Sec.TpduVerify.Request, le TSC commence les étapes appropriées de traitement des TPDU pour vérifier et, si configuré, déchiffrer la TPDU reçue comme dicté par les étapes de traitement pour les TPDU reçues en 7.3.3.9.

7.3.3.5.9 Sec.TpduVerify.Response

7.3.3.5.9.1 Généralités

Sec.TpduVerify.Response rend compte du résultat d'une Sec.TpduVerify.Request.

7.3.3.5.9.2 Sémantique

La sémantique de Sec.TpduVerify.Response est comme suit:

```
Sec.TpduVerify.Response    (
    TSDU,
    TSDU_Size,
    Time_Of_TPDU_Creation,
    Status)
```

Le Tableau 55 spécifie les éléments pour Sec.TpduVerify.Response.

Tableau 55 – Eléments de Sec.TpduVerify.Response

Nom de l'élément	Identificateur de l'élément	Type de l'élément scalaire
TSDU (après un déchiffrement exigé quelconque)	1	Type: OctetString
TSDU_Size	2	Type: Unsigned Plage valide: 0..Assigned_Max_TSDU_Size; voir 11.4.3.3
Time_Of_TPDU_Creation (représentation fractionnaire à point fixe sur 32 bits du temps TAI, modulo 2^{22} s, utilisée dans le nonce)	3	Type: Unsigned32
Status	4	Type: Unsigned Valeurs nommées: 0: réussite; > 0: échec

7.3.3.5.9.3 Moment de sa génération

Le TSC génère la Sec.TpduVerify.Response en réponse à une Sec.TpduVerify.Request. La Sec.TpduVerify.Response retourne une valeur de statut qui indique soit SUCCESS, soit le code d'erreur approprié.

7.3.3.5.9.4 Utilisation appropriée

A la réception de Sec.TpduVerify.Response, la TL reçoit notification du résultat de la vérification et du déchiffrement éventuel de la TPDU entrante.

7.3.3.6 Structure de l'en-tête de sécurité de TPDU

La structure de l'en-tête de sécurité de TPDU est telle que décrite dans le Tableau 56.

Tableau 56 – Structure de l'en-tête de sécurité de TL

Octet	Bits							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	Security_Control							
2 (opt)	Crypto_Key_Identifier							
3	Nominal_Time							
4								

L'en-tête de sécurité de TL doit être ajouté à toutes les TPDU pour utiliser la compression d'en-tête dans la NL. Dans le cas où aucun KeyDescriptor ne correspond à une TPDU spécifique, cette TPDU doit être traitée comme si elle comportait un niveau de sécurité égal à NONE.

NOTE 1 Lorsque la TPDU n'a aucun TMIC, la somme de contrôle UDP est utilisée pour la détection d'erreurs.

Les champs comprennent:

- Security_Control: comme défini dans le 7.3.1.2.
- Crypto_Key_Identifier: spécifie le CryptoKeyIdentifier courant utilisé pour protéger cette TPDU.
- Nominal_Time: la partie de temps doit être égale à 16 bits du temps TAI, exprimé en 2^6 s en unités de 2^{-10} s, présentée dans l'ordre du bit de poids fort (MSB).

NOTE 2 Le fait de fixer la granularité du temps à 2^{-10} s donne à la TL la capacité d'émettre 1 023 TPDU par seconde. Avec la taille maximale de charge utile d'une TPDU, cela est adéquat pour le débit spécifié par 6LoWPAN. Une granularité variable pour le temps de TPDU, appropriée pour prendre en charge des processus d'automation de haut débit, est un domaine possible d'une normalisation future.

7.3.3.7 Construction du nonce pour les TPDU

La présente norme utilise une construction de nonce de TPDU différente (mais connexe) de celle de ses DPDU. Un nonce de 13 octets est exigé pour le moteur CCM*. Le nonce doit être construit comme une concaténation allant du premier (le plus à gauche) octet au dernier (le plus à droite) octet des champs de données conformément au Tableau 57, où:

- l'adresse EUI64Address doit être utilisée comme une matrice de 8 octets et le temps TAI tronqué;
- le temps TAI nominal de la création de la TPDU doit être réglé à une granularité de 2^{-10} s, et ne doit pas être plus précoce de plus de 1 s que le temps local réel du début de la création de la TPDU, et ne doit pas être plus tardif de plus de 1 s que le temps local réel de la fin de la création de la TPDU. Chaque TPDU sortante provenant d'une adresse EUI64Address source spécifique qui utilise une clé donnée doit être créée avec une valeur unique pour le temps TAI nominal de création de TPDU tronqué sur 32 bits. Ce codage restreint la fréquence maximale de la TL à 1 024 TPDU par seconde qui ne doit pas être dépassée. La structure du temps TAI nominal tronqué de 32 bits doit être telle que décrite dans le Tableau 58.

Tableau 57 – Structure du nonce de TPDU

Octet	Bits							
	7	6	5	4	3	2	1	0
1	EUI64Address							
...								
8								
9	Temps TAI nominal tronqué de création de TPDU							
...								
12								
13	0xFF							

La représentation nominale tronquée sur 32 bits du temps TAI utilisée dans le nonce T est décrite dans le Tableau 58.

Tableau 58 – Structure du temps TAI nominal tronqué de 32 bits utilisé dans le nonce T

Octet	Bits							
	7	6	5	4	3	2	1	0
1	Temps TAI nominal (bits avec un poids de $2^{21}..2^{14}$ s)							
2	Temps TAI nominal (bits avec un poids de $2^{13}..2^{6}$ s)							
3	Temps TAI nominal (bits avec un poids de $2^{5}..2^{-2}$ s)							
4	Temps TAI nominal (bits avec un poids de $2^{3}..2^{-10}$ s)							

Pour une granularité de 2^{-10} s, il y aura 48,5 jours avant que la représentation du temps de 32 bits ne se répète, fournissant de ce fait tout au plus 48,5 jours avant qu'une nouvelle clé n'ait besoin d'être déployée pour éviter une collision potentielle de nonce avec la réutilisation résultante de keystream.

NOTE Cette représentation est choisie parce que l'expéditeur et les récepteurs prévus sont présumés partager approximativement le même sens du temps et le même temps de début nominal pour n'importe quel intervalle de temps MAC.

7.3.3.8 Traitement pour les TPDU devant être émises

Les entrées à la procédure de sécurité pour les TPDU devant être émises sont:

- la TPDU devant être sécurisée;
- l'adresse EUI64Address de l'appareil source;
- le temps TAI nominal;
- les adresses IPv6Address source et de destination; et
- le port source et de destination.

Les sorties délivrées par cette procédure sont:

- le statut de la procédure; et
- si ce statut est SUCCESS, la TPDU sécurisée.

La procédure de sécurité pour les TPDU construites pour l'émission est constituée des étapes suivantes:

a) La procédure doit obtenir le KeyDescriptor issu du Tableau 93 satisfaisant aux critères de sélection suivants:

   1) Les entrées avec Type = 10 (TL). Si aucune n'est disponible, la procédure doit retourner avec un statut UNAVAILABLE_KEY.

2) De ces entrées, les entrées avec le KeyLookupData concordant à SourceAddress||SourcePort||DestinationAddress||DestinationPort (voir Tableau 94) pour cette TPDU. Si aucun KeyDescriptor n'est disponible et les deux conditions suivantes sont toutes les deux vraies, la procédure doit traiter la TPDU comme étant une TPDU sans sécurité (niveau de sécurité= NONE). Autrement, la procédure doit retourner avec un statut UNAVAILABLE_KEY.

3) Condition1: L'état de rattachement de l'appareil de réception est Provisioned ou Joining (voir Tableau 79).

4) Condition2: Les ports source et de destination sont tous deux pour le DMAP (à savoir 0xF0B0).

Ces conditions ont besoin d'être respectées pour émettre une TPDU de rattachement qui a un niveau de sécurité de NONE.

5) De ces entrées, les entrées valides pour la période courante, satisfaisant à l'inégalité "ValidNotBefore < temps courant < ValidNotAfter", doivent être sélectionnées. Si aucune n'est disponible, la procédure doit retourner avec un statut UNAVAILABLE_KEY.

6) De ces entrées, si au moins deux clés sont valides pour le temps courant, la procédure doit sélectionner la clé avec la plus longue valeur de ValidNotAfter.

7) De ces entrées, si au moins deux clés ont le même ValidNotAfter, la procédure doit sélectionner la clé avec le plus petit ValidNotBefore.

8) De ces entrées, si au moins deux clés ont le même ValidNotBefore, la procédure doit sélectionner la clé avec le CryptoKeyIdentifier le plus élevé.

Si la procédure échoue, la procédure doit traiter la TPDU comme étant sans sécurité (niveau de sécurité = NONE).

b) La procédure doit récupérer la politique dans le KeyDescriptor sélectionné.

c) La procédure doit déterminer si la TPDU devant être sécurisée satisfait à la contrainte relative à la taille maximale des TPDU, comme suit:

1) La procédure doit établir la taille M, en octets, du champ d'authentification de TMIC à partir du niveau de sécurité.

2) La taille du champ Key Index dans l'en-tête de sécurité de TL doit être 1 octet, si plus de 1 clé est valide pour l'association de sécurité courante et 0 autrement.

3) La procédure doit déterminer l'expansion de données comme étant la taille de CryptoKeyIdentifier + M.

4) La procédure doit vérifier si la taille de la TPDU devant être sécurisée, y compris l'expansion de données, est inférieure ou égale à Assigned_Max_TSDU_Size (voir Tableau 30). Si cette vérification échoue, la procédure doit retourner un statut de TPDU_TOO_LONG.

d) La procédure doit construire l'octet de contrôle de sécurité de l'en-tête de sécurité de TL. Si le niveau de sécurité concorde avec plus d'un KeyDescriptor issu du Key Descriptor courant, le CryptoKeyIdentifier doit être utilisé avec le CryptoKeyIdentifierMode = 0x01; autrement, la procédure doit mettre le CryptoKeyIdentifierMode = 0x00.

e) La procédure doit mettre le CryptoKeyIdentifier = CryptoKeyIdentifier issu du courant Key Descriptor (s'il est présent) dans l'en-tête de sécurité de TL. Voir Tableau 56.

f) La procédure doit construire le temps TAI nominal au format TAITimeRounded décrit dans le Tableau 58.

g) La procédure doit régler les octets de Nominal_Time dans l'en-tête de sécurité de la TL sortante comme étant les derniers 16 bits de la valeur du temps TAI nominal au format TAITimeRounded. Voir octets 3 et 4 dans le Tableau 58.

h) Si aucun Key Descriptor n'a été trouvé, aller à l'étape j); autrement, la procédure doit utiliser l'adresse EUI64Address, le temps TAI nominal au format TAITimeRounded et la valeur 0xFF de 8 bits pour construire le nonce conformément au Tableau 57.

i) La procédure doit utiliser le nonce, le support de clé, l'en-tête de TPDU, la charge utile de TPDU et le mode CCM* de fonctionnement tels que décrits dans l'IEEE 802.15.4:2011, 7.3.4, pour sécuriser la TPDU:

   1) Si le paramètre SecurityLevel spécifie l'utilisation du chiffrement (voir IEEE 802.15.4:2011, Tableau 58), l'opération de chiffrement doit être appliquée seulement au champ de charge utile de la TPDU. Le champ de charge utile correspondant est passé au processus de transformation CCM* décrit dans l'IEEE 802.15.4:2011, 7.3.4, comme étant la charge utile non sécurisée. La charge utile chiffrée résultante doit être substituée à la charge utile d'origine.

   2) Les champs restants dans la TPDU, jusqu'au champ de charge utile non compris, plus tous les éventuels champs virtuels exigés, doivent être passés au processus de transformation CCM* décrit dans l'IEEE 802.15.4:2011, 7.3.4, comme étant le champ non-charge utile

   3) L'ordonnancement et la manière exacte d'accomplir les opérations relatives au chiffrement et à l'intégrité ainsi que le placement des données chiffrées ou du code d'intégrité ainsi obtenus au sein du champ de charge utile de la TPDU doivent être tels que définis dans l'IEEE 802.15.4:2011, 7.3.4.

j) La procédure doit retourner la TPDU sécurisée et un statut de SUCCESS.

7.3.3.9 Traitement pour les TPDU reçues

L'entrée à la procédure de sécurité pour les TPDU reçues est la TPDU devant être dépourvue de sécurité, qui contient les adresses IPv6Address source et destination et les ports source et de destination. Les sorties délivrées par cette procédure sont la TPDU non sécurisée, le niveau de sécurité, le CryptoKeyIdentifierMode, la source de clé, l'indice de clé et le statut de la procédure. Toutes les sorties de cette procédure sont supposées être non valides, jusqu'à ce que cela soit explicitement établi dans cette procédure. Chaque destinataire de TPDU maintient un cache des valeurs de nonce authentifiées des TPDU récemment reçues.

La procédure de sécurité à la réception des TPDU est constituée des étapes suivantes:

a) La procédure doit obtenir le niveau de sécurité et le CryptoKeyIdentifierMode issus des sous-champs correspondants du champ contrôle de sécurité et l'indice de clé issu des sous-champs correspondants du CryptoKeyIdentifier (s'il est présent) de l'en-tête de sécurité de la TPDU entrante.

b) La procédure doit reconstruire le temps TAI nominal d'initiateur de la formation de la TPDU qui est inféré (Voir Note 2).

c) La procédure doit comparer le temps dans l'étape b) et le temps TAI courant du destinataire. Si le temps dans l'étape b) est en avance de plus de 2 s par rapport au temps TAI courant du destinataire, ou en retard de plus de N secondes par rapport au temps TAI courant du destinataire (où N est un paramètre déterminé par une politique dont la valeur par défaut est 62 s), le processus de sécurité retourne FAILURE_TPDU_DID_NOT_AUTHENTICATE.

d) La procédure doit obtenir le KeyDescriptor issu du Tableau 93 satisfaisant aux critères de sélection suivants:

   1) Les entrées avec Type = 10 (TL).

   2) De ces entrées, les entrées avec le KeyLookupData concordant à SourceAddress||SourcePort||DestinationAddress||DestinationPort (voir Tableau 94) pour cette TPDU. Si aucun KeyDescriptor n'est disponible et les deux conditions suivantes sont toutes vraies, la procédure doit traiter la TPDU comme étant une TPDU sans sécurité (niveau de sécurité= NONE). Autrement, la procédure doit retourner avec un statut UNAVAILABLE_KEY.

      i) Condition1: L'état de rattachement de l'appareil de réception est Provisioned ou Joining (voir Tableau 79).

      ii) Condition2: Les ports source et de destination sont tous deux pour les DMAP (à savoir 0xF0B0).

NOTE 1 Cette information est utilisée lors du traitement d'une TPDU de rattachement reçue qui a un niveau de sécurité de NONE.

3) De ces entrées, les entrées valides pour la période courante, satisfaisant à l'inégalité "ValidNotBefore < temps courant < ValidNotAfter", doivent être sélectionnées. Si aucune n'est disponible, la procédure doit retourner avec un statut UNAVAILABLE_KEY.

4) De ces entrées, si au moins deux clés sont valides pour le temps courant, la procédure doit sélectionner la clé avec la plus longue valeur de ValidNotAfter.

5) De ces entrées, si au moins deux clés ont le même ValidNotAfter, la procédure doit sélectionner la clé avec le plus petit ValidNotBefore.

6) De ces entrées, si au moins deux clés ont le même ValidNotBefore, la procédure doit sélectionner la clé avec le CryptoKeyIdentifier le plus élevé.

7) Si la procédure échoue, la procédure doit retourner avec un statut UNAVAILABLE_KEY.

e) La procédure doit déterminer si le niveau de sécurité de la TPDU entrante se conforme à la politique de niveau de sécurité en comparant le SecurityLevel du Key Descriptor concordant obtenu dans l'étape b) ci-dessus. S'il y a une discordance, la procédure doit retourner avec un statut IMPROPER_SECURITY_LEVEL.

f) La procédure doit alors utiliser l'adresse EUI64Address de l'initiateur, le temps TAI nominal de formation de TPDU issu de l'étape b), les 16 bits de poids faible du temps TAI nominal (voir Tableau 58) reçus, pour générer le nonce conformément au Tableau 57.

g) La procédure doit utiliser le nonce, la clé issue du KeyDescriptor obtenu dans l'étape d), les en-têtes réels (les champs non-charge utile), la charge utile et le MIC de la TPDU entrante et le mode de fonctionnement CCM* tels que décrits dans les opérations (voir l'IEEE 802.15.4:2011, 7.3.5) pour authentifier et, si configuré pour l'association de transport, déchiffrer la TPDU:

1) Si le niveau de sécurité spécifie l'utilisation du chiffrement (voir l'IEEE 802.15.4:2011, Tableau 58), l'opération de déchiffrement doit être appliquée seulement au champ charge utile réel de la TPDU (voir l'IEEE 802.15.4:2011, 5.2.2.2.2). Le champ de charge utile correspondant doit être passé au processus de transformation inverse CCM* décrit dans l'IEEE 802.15.4:2011, 7.3.5, comme étant la charge utile sécurisée.

2) Les champs restants dans la TPDU, plus tous les éventuels champs virtuels exigés, doivent être passés au processus de transformation inverse CCM* décrit dans l'IEEE 802.15.4:2011, 7.3.5, comme étant les champs non-charge utile (voir IEEE 802.15.4:2011, Tableau 57).

3) L'ordonnancement et la manière exacte d'accomplir les opérations relatives au déchiffrement et à la vérification d'intégrité ainsi que le placement des données déchiffrées ainsi obtenues au sein du champ de charge utile de la TPDU doivent être tels que définis dans l'IEEE 802.15.4:2011, 7.3.5.

h) Si le processus de transformation inverse CCM* échoue, alors la procédure peut décrémenter de 64 s le temps TAI nominal et répéter le processus ci-dessus pendant la période DSMO.pduMaxAge. Autrement, la procédure doit établir la TPDU devant être dépourvue de sécurité et retourner un statut de SECURITY_ERROR.

i) La procédure doit rechercher par consultation le nonce de la TPDU juste authentifiée dans le cache, avec les résultats possibles suivants:

1) Le temps du nonce est plus ancien que le temps le plus ancien dans le cache et le cache n'a pas de place pour une autre entrée plus ancienne, donc le processus de sécurité retourne FAILURE_OVERAGE_TPDU.

2) Le nonce est déjà dans le cache, donc le processus de sécurité retourne FAILURE_DUPLICATE_TPDU.

3) Toute charge utile chiffrée de la TPDU est déchiffrée, et le processus de sécurité retourne SUCCESS.

j) La procédure doit insérer la valeur de nonce dans le cache, au besoin en expulsant du cache l'entrée de cache ayant le temps TAI nominal inféré le plus ancien de formation de TPDU et retourner avec la TPDU dépourvue de sécurité, le niveau de sécurité, le

CryptoKeyIdentifierMode, la source de clé, l'indice de clé (si présent) et un statut de SUCCESS.

NOTE 2 Le temps TAI nominal de formation de TPDU de l'initiateur est initialement inféré à partir du temps TAI courant du destinataire et du temps TAI nominal fractionnaire de création de TPDU qui est spécifié dans la TPDU, en satisfaisant à la relation

((current-receiver-time + 2 s) ≥ originator's-nominal-time ≥ (current-receiver-time – DSMO.pduMaxAge)).

La durée de 2 s est prévue pour couvrir ±1 s plus les conditions aux limites.

Il est permis que le cache soit segmenté en des caches séparés pour chaque adresse EUI64Address d'envoi. Il est en plus permis que le cache soit segmenté par la QoS de couche réseau rapportée, afin qu'un cache ne contenant que quelques nonces de TPDU de faible priorité puisse aussi ne pas contenir des douzaines à des centaines de nonces pour devancer des TPDU de plus haute priorité. Il est également permis que la taille du cache soit adaptative, afin que les occurrences répétées du premier résultat de l'étape g) ci-dessus fasse croître le cache, avec la réduction appropriée de la taille du cache au cas où la capacité du cache en excès n'aurait pas été utilisée pendant une période prolongée.

7.3.3.10 Détection et rejet des TPDU dupliquées ou rejouées

Voir 7.3.3.9, i).

7.4 Processus de rattachement

7.4.1 Généralités

Le processus de rattachement décrit les étapes par lesquelles un nouvel appareil est admis dans un réseau conforme à une norme et obtient toutes les informations pertinentes pour être capable de communiquer avec d'autres appareils ainsi qu'avec le gestionnaire de système et le gestionnaire de sécurité.

NOTE Cette description suppose que l'appareil qui se rattache a une pile de protocoles DL conforme à l'une des éditions de la présente norme. Cependant, comme cette procédure est un protocole d'AL, elle est également utilisable pour les appareils qui ne comportent pas de pile de DL en omettant simplement les étapes relatives à la DL.

7.4.2 Conditions préalables

Le processus de rattachement suit l'étape de configuration, pendant laquelle les informations cryptographiques et les paramètres de configuration non cryptographiques peuvent être fournis au nouvel appareil. Un nouvel appareil doit obtenir de l'appareil de configuration ces informations nécessaires relatives à la configuration. Cela est décrit à l'Article 13. L'attribut Join_Command dans le DMO d'un appareil doit être utilisé pour ordonner à l'appareil de rejoindre le réseau.

Un appareil se rattachant doit rejoindre le réseau cible en utilisant l'une des approches de sécurité suivantes:

- clés symétriques;
- clés asymétriques;
- sans sécurité.

L'approche sans sécurité n'utilise pas une clé secrète pour le transfert de clés de rattachement. Au lieu de cela, elle utilise l'une des clés prédéfinies bien connues K_global ou K_open, telles que spécifiées en 7.2.2.2. Dans ce cas, le MIC fonctionne comme un CRC fort, qui n'offre aucune assurance de sécurité, mais a une probabilité très élevée de détection d'erreurs qui ne sont pas dues à une attaque délibérée. Dans ce cas, les sessions sécurisées de bout à bout (associations T) ne sont pas autorisées.

Un appareil mettant en œuvre l'approche de rattachement à clés symétriques doit avoir à la fois une clé de rattachement symétrique et l'adresse EUI64Address d'un gestionnaire de sécurité qui partage cette clé de rattachement.

Un appareil mettant en œuvre l'approche de rattachement à clés asymétriques doit avoir un certificat signé par une autorité de certification ayant la confiance du réseau-cible.

Un appareil mettant en œuvre l'approche de rattachement sans sécurité doit avoir la clé symétrique publiée, bien connue et non secrète commune à tous les réseaux conformes à une norme, K_global or K_open, telle que spécifiée en 7.2.2.2.

7.4.3 Etat final et propriétés souhaités de l'appareil

A la fin du processus de rattachement, le système doit avoir l'état suivant:

- le nouvel appareil et le gestionnaire de sécurité partagent en toute sécurité une clé principale symétrique de long terme;
- si une DLE de WISN est présente dans l'appareil, le nouvel appareil a le support cryptographique exigé pour que cette DLE échange des DPDU avec ses voisins directs;
- si une DLE WISN est présente dans l'appareil, le nouvel appareil a le support non cryptographique et les ressources exigés pour que cette DLE échange des DPDU avec au moins l'un de ses voisins directs; et
- le nouvel appareil doit avoir un contrat avec le gestionnaire de système.

NOTE 1 Une contrat avec le gestionnaire de système inclut une clé T partagée entre le gestionnaire de système et une TLE du nouvel appareil.

Lors de l'utilisation de l'approche à clés symétriques ou à clés asymétriques, le processus de rattachement fournit les assurances de sécurité suivantes:

- la protection contre les attaques de rejet sur les APDU de rattachement:
  - l'assurance cryptographique au nouvel appareil que le gestionnaire de sécurité est en vie;
  - l'assurance cryptographique au gestionnaire de sécurité que le nouvel appareil est en vie;
- authenticité:
  - assurance cryptographique que la demande de rattachement vient d'un appareil qui a le matériel de confiance valide;
  - assurance cryptographique que les APDU de rattachement n'ont pas été altérées;
- confidentialité:
  - protection cryptographique pour les clés dans la réponse de rattachement, telle qu'une oreille indiscrète ne puisse pas récupérer les clés transportées.

NOTE 2 Un défi ("challenge")/protocole de réponse est utilisé pour que le processus de rattachement sécurisé élimine toute nécessité de se fier, au moment du processus de rattachement, à une source de temps TAI de confiance mutuelle.

7.4.4 Etapes de processus de rattachement communes aux approches à clés symétriques et à clés asymétriques

7.4.4.1 Généralités

Lors de l'utilisation de l'approche sécurisée à clés symétriques ou à clés asymétriques, l'appareil passe par les étapes générales suivantes pour parachever le processus de rattachement.

Le gestionnaire de système commande le traitement d'un nouvel appareil rejoignant le réseau. Un appareil non rattaché qui met en œuvre une DLE conforme à la présente norme se met à l'écoute pour détecter des DPDU d'annonce issues de routeurs locaux, dont les fonctions d'annonce sont configurées par le gestionnaire de système.

Des annonces peuvent être trouvées en utilisant le balayage actif, le balayage passif, ou une combinaison des deux. Le balayage actif implique des DPDU de sollicitation envoyées par l'appareil se rattachant pour demander l'émission de DPDU d'annonce. Des informations détaillées pour le balayage actif/passif sont données en 9.1.13.

Un tel routeur d'annonce doit apporter une assistance pendant le processus de rattachement en agissant comme un proxy pour un gestionnaire de système, relatif au nouvel appareil. Comme proxy, ce routeur d'annonce transmet la demande de rattachement issue du nouvel appareil vers un gestionnaire de système et transmet la réponse de rattachement issue du gestionnaire de système en question vers le nouvel appareil. A la réception d'une demande de rattachement issue d'un nouvel appareil, un gestionnaire de système traite la demande, authentifie l'acceptabilité de la demande par une communication avec un gestionnaire de sécurité et génère en réponse une réaction de rattachement.

7.4.4.2 Construction des PDU du processus de rattachement

La demande de rattachement issue d'un nouvel appareil est constituée de PDU concaténées qui séparent les informations de sécurité, échangées entre l'appareil et un gestionnaire de sécurité, des informations autres que de sécurité, échangées entre l'appareil et un gestionnaire de système. La réponse de rattachement est constituée de semblables PDU concaténées qui séparent les informations de sécurité des informations autres que de sécurité.

Les informations autres que de sécurité échangées au cours du processus de rattachement sont décrites en 6.3.9.2. Cet échange utilise la méthode définie en 6.3.9.2.2 pour les DMO du routeur d'annonce et les méthodes définies en 6.3.9.5 pour le DMSO du gestionnaire de système.

Les informations de sécurité échangées au cours du processus de rattachement dépendent de l'approche de rattachement utilisée, telle que décrite en 7.4.5 pour l'approche à clés symétriques et en 7.4.6 pour l'approche à clés asymétriques.

Afin que le nouvel appareil construise ses PDU de processus de rattachement et les envoie au routeur d'annonce, il a besoin de connaître l'adresse EUI64Address du routeur d'annonce. Le processus suivi par le nouvel appareil pour obtenir cette adresse EUI64Address est décrit en 9.1.14. Les PDU de demande de processus de rattachement, du nouvel appareil vers le routeur d'annonce, et les PDU de réponse de processus de rattachement, du routeur d'annonce vers le nouvel appareil, doivent être construites comme suit, en utilisant cette adresse EUI64Address du routeur d'annonce et l'adresse EUI64Address du nouvel appareil:

- S'il y a une DLE de WISN présente dans l'appareil se rattachant, l'en-tête de DL pour ces PDU de processus de rattachement est construit tel que décrit en 9.3.
- L'en-tête de NL pour ces PDU de processus de rattachement est construit tel que décrit en 10.5.3.
- L'en-tête de TL pour ces PDU de processus de rattachement est construit tel que décrit en 11.5. Pour le calcul de la somme de contrôle UDP, le pseudo-en-tête UDP pour IPv6 utilise l'adresse EUI64Address du nouvel appareil et l'adresse EUI64Address du routeur d'annonce, représentées comme étant les adresses IPv6Address locales à une liaison des appareils respectifs, comme les adresses IPv6Address de la source et de la destination des PDU.
- Les DMAP de l'appareil se rattachant et du routeur d'annonce utilisent ces adresses IPv6Address locales à une liaison lors de l'acheminement des PDU relatives à la demande de rattachement vers leurs TLE.

- A la TL, la Sec.TpduOutCheck.Response doit retourner avec une valeur de 3 pour la taille d'en-tête de sécurité et de 0 pour la taille de TMIC, indiquant un en-tête de sécurité de TL avec le niveau de sécurité = 0 (NONE). En raison de l'absence (habituelle) de clés secrètes partagées, la protection de sécurité de TL n'est généralement pas disponible pour l'échange de TPDU des PDU de demande de rattachement issues nouvel appareil et des PDU de demande de réponse issues du routeur d'annonce.
- S'il y a une DLE présente, l'en-tête de sécurité de DPDU a un niveau de sécurité = 1 (MIC-32), utilisant donc un DMIC de 32 bits pour les DPDU de rattachement, qui doivent être construites telles que décrites en 7.3.2.5 en utilisant la clé D K_global (CryptoKeyIdentifier = 0). Parce que cette clé est bien connue, elle n'assure aucune protection contre les attaques délibérées. Par conséquent, ce DMIC de 32 bits n'est utilisé que pour détecter des erreurs involontaires dans les DPDU de demande et de réponse de rattachement. Le routeur D d'annonce doit utiliser son contrat existant avec le gestionnaire de système pour transmettre les PDU de processus de rattachement pour le compte de l'appareil qui fait la demande de rattachement.

NOTE Un nouvel appareil qui essaie de rejoindre le réseau n'a aucun contrat assigné par le gestionnaire de système. Par conséquent, la communication entre le nouvel appareil et le routeur d'annonce n'est basée sur aucun contrat.

Il convient que les messages de demande et de réponse de PSMO.Security_Sym_Confirm() soient protégés par des informations de clé D et de clé T qui sont distribuées dans le message de réponse de PSMO.Proxy_Security_Sym_Join().

7.4.4.3 Protection des messages de processus de rattachement

7.4.4.3.1 Généralités

Comme le nouvel appareil n'a pas la clé de sous-réseau D nécessaire et une clé T de niveau de TL avec le routeur d'annonce, tous les messages de processus de rattachement, autres que les messages de confirmation, entre le nouvel appareil et le routeur d'annonce doivent utiliser la K_global au niveau de la DL pour construire un DMIC de 32 bits. Au niveau de la TL, la somme de contrôle UDP doit être utilisée pour ces messages.

Les informations de sécurité dans la demande de rattachement, ainsi que toutes les informations revenant du gestionnaire de système et du gestionnaire de sécurité dans les messages de réponse de rattachement, sont protégées en utilisant la clé de rattachement. Cela est décrit en 7.4.5 pour l'approche à clés symétriques et en 7.4.6 pour l'approche à clés asymétriques.

7.4.4.3.2 Protection contre les attaques de rejet des PDU de rattachement

Pour protéger contre une attaque de rejet de PDU de rattachement, il convient que le gestionnaire de sécurité effectue une vérification pour détecter des doublons de défis avec un MIC valide issu du nouvel appareil. Si aucun doublon de défi n'est détecté, le gestionnaire de sécurité stocke la valeur de défi en vue d'une vérification ultérieure de doublons. En cas de détection d'un doublon, le gestionnaire de sécurité rejette la PDU avant de traiter la PDU de rattachement.

7.4.4.3.3 Protection d'un message sans sécurité dans le processus de rattachement

7.4.4.3.3.1 Généralités

Le paragraphe 7.4.4.3.3 décrit la configuration des valeurs de réglage de sécurité au cours du processus de rattachement. Les informations réseau non relatives à la sécurité sont configurées avec les méthodes DMSO.System_Manager_Join() et DMSO.System_Manager_Contract(). Les détails des méthodes sont spécifiés en 6.3.9.2. De tels messages non relatifs à la sécurité sont protégés avec des opérations cryptographiques à l'AL. Les messages non relatifs à la sécurité sont générés dans le gestionnaire de système et passés au gestionnaire de sécurité qui additionne alors le MIC en utilisant la clé de

rattachement. Les messages protégés sont émis vers l'appareil se rattachant, par l'intermédiaire du DMSO dans le gestionnaire de système. A l'appareil se rattachant, le MIC dans la réponse reçue est validé avec la même opération.

A l'appareil se rattachant, le MIC dans la réponse reçue est validé avec la même opération.

7.4.4.3.3.2 Génération de MIC pour la réponse de System_Manager_Join

La méthode DMSO.System_Manager_Join() est définie en 6.3.9.5. Le champ MIC est constitué des 4 octets de poids fort dans le MACTag généré par l'opération suivante:

MACTag = HMAC-MMOK_join[Output Argument number1 .. number6 in Tableau 23 || $\text{EUI64Address}_{\text{join\_device}}$ || $\text{Challenge}_{\text{join\_device}}$].

7.4.4.3.3.3 Génération de MIC pour la réponse System_Manager_Contract

La méthode DMSO.System_Manager_Contract est définie en 6.3.9.5. Le champ MIC est constitué des 4 octets de poids fort dans le MACTag généré par l'opération suivante:

MACTag = HMAC-MMOK_join[Output Argument number1 in Tableau 24 || $\text{EUI64Address}_{\text{join\_device}}$ || $\text{Challenge}_{\text{join\_device}}$]

7.4.4.3.3.4 Confirmation

Après la réception de DMO.Proxy_System_Manager_Join().Response et DMO.Proxy_System_Manager_Contract().Response, l'appareil se rattachant envoie un message pour informer que les informations réseau correctes ont été reçues par le gestionnaire de système.

Dans le processus de rattachement à clé symétrique, le processus de confirmation est intégré dans la méthode PSMO.Security_Confirm() spécifiée dans le Tableau 61.

Dans le processus de rattachement à clés asymétriques, le processus de confirmation est accompli avec la méthode PSMO.Network_Information_Confirmation() spécifiée dans le Tableau 74.

7.4.4.4 Temporisateurs de rattachement

Dans le processus de rattachement, deux temporisateurs sont définis. A l'expiration de l'un de ces temporisateurs, toute information (par exemple, état et paramètre reçu) placée en cache pour le processus de rattachement particulier doit être enlevée ou réinitialisée.

JT_1: Durée temporelle gérée dans l'appareil se rattachant, depuis le temps d'émission de la demande de rattachement de sécurité jusqu'au temps de validation correcte de la réponse de confirmation générée par le gestionnaire de sécurité. Si l'appareil se rattachant n'est pas un appareil dorsal, la valeur initiale pour JT_1 doit être mise à la valeur de DauxJoinTimeout qui est distribuée au sein de l'annonce de DL (voir Tableau 127). Autrement, la valeur réelle pour JT_1 doit être mise à 60 s pour l'appareil dorsal.

JT_2: Durée temporelle gérée dans le gestionnaire de sécurité, depuis le temps de réception de la demande de rattachement de sécurité jusqu'au temps de validation correcte du message de confirmation de sécurité générée par l'appareil se rattachant. La valeur réelle de JT_2 n'est pas spécifiée dans la présente norme.

NOTE JT_2 peut être inférieur à JT_1.

7.4.4.5 Processus de rattachement d'appareil dorsal

Un appareil dorsal rejoint un réseau cible en exécutant la méthode de rattachement dans le DMO du gestionnaire de système au lieu de celui du routeur d'annonce. Par conséquent, l'appareil dorsal n'a pas besoin de découvrir un routeur d'annonce; la

DPO.Target_System_Manager_Address doit être mise dans la phase de configuration. La vue d'ensemble du processus de rattachement de l'appareil dorsal est illustrée à la Figure 45 pour le processus de rattachement à clés symétriques et à la Figure 48 pour le processus de rattachement à clés asymétriques.

7.4.4.6 Contraintes de taille de TMIC pour la session entre le nœud de rattachement et le gestionnaire de système

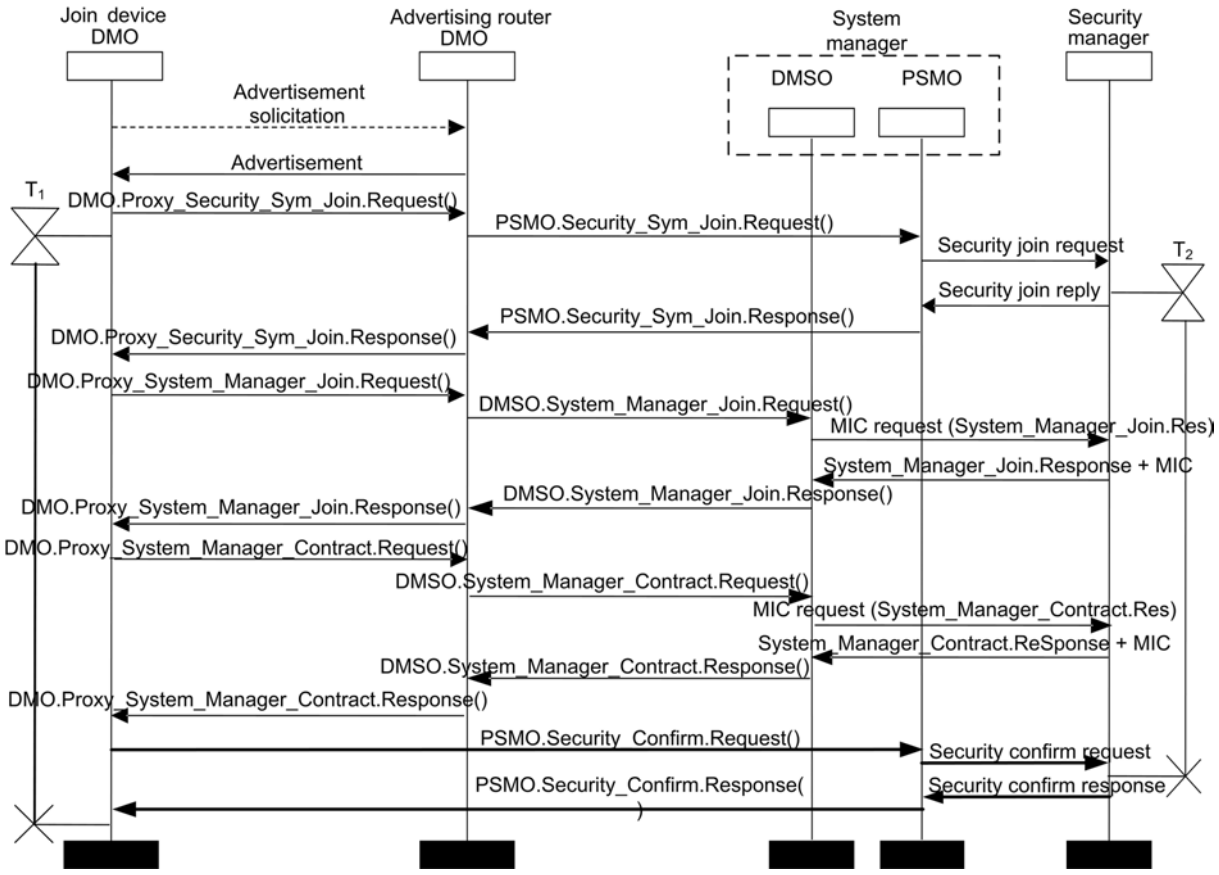
A la fin du processus de rattachement, le gestionnaire de sécurité assigne un niveau de sécurité initial pour la session entre l'appareil se rattachant et le gestionnaire de système. Ce niveau de sécurité doit être de 0, 1, 2, 5 ou 6; il ne doit pas être égal à 3 (MIC-128) ni 7 (ENC-MIC-128), la valeur 4 (ENC-only) étant toujours invalide.

7.4.5 Processus de rattachement à clés symétriques

7.4.5.1 Généralités

La Figure 44 illustre la messagerie impliquée dans le processus de rattachement à clés symétriques par lequel un nouvel appareil doit rejoindre un réseau en fonctionnement dans lequel il n'a pas été récemment participant. Le flux montre le cas normal dans lequel il ne se produit pas d'erreur ou de temporisation. Les temporisations sont spécifiées au Tableau 92.

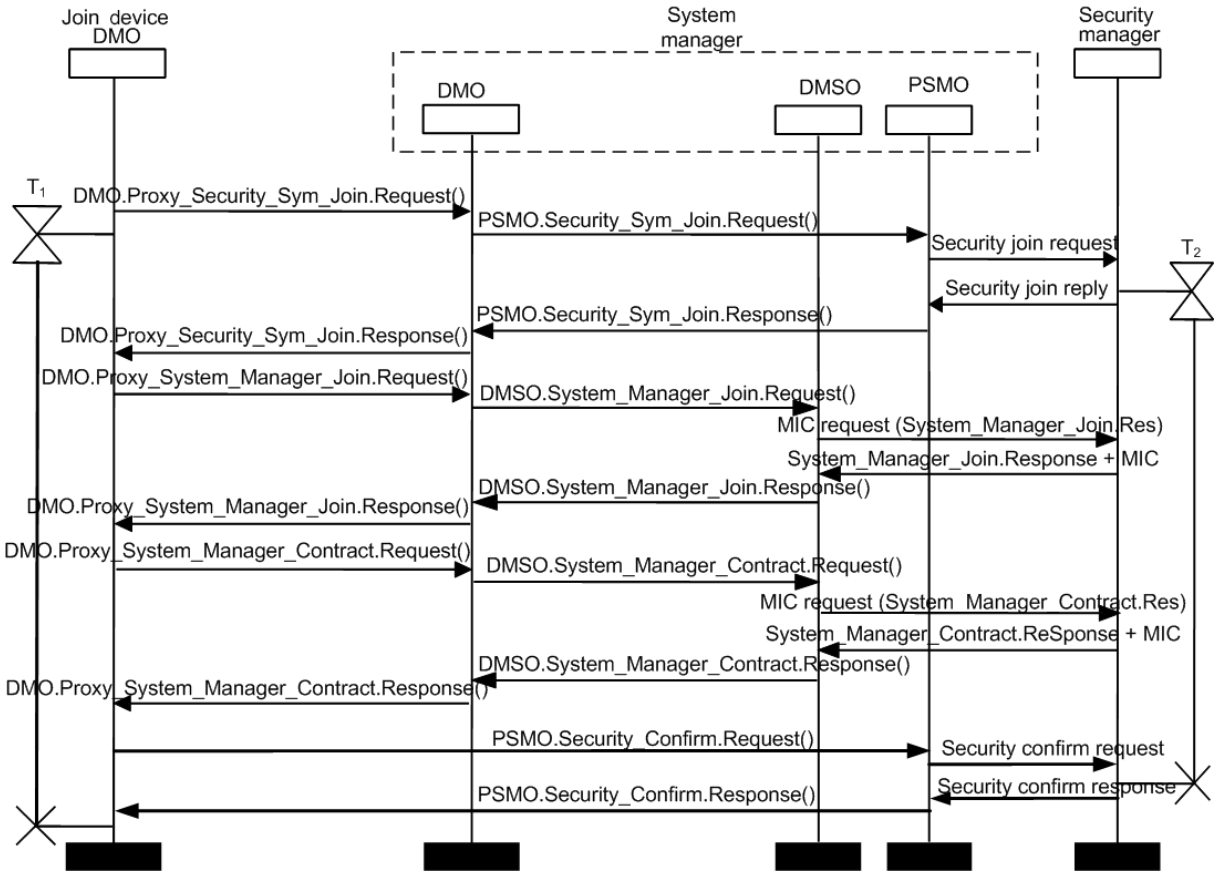
Sur l'appareil se rattachant, le processus de rattachement à clés symétriques doit être initié avec une DMO.Proxy_Security_Sym_Join().Request et finalisé avec une PSMO.Security_Confirm().Response valide. Sur le gestionnaire de sécurité, le processus de rattachement à clés symétriques doit être initié avec un message valide dérivé de PSMO.Security_Sym_Join().Request et finalisé avec un message valide dérivé avec PSMO.Security_Confirm().Request.



Légende

Anglais	Français
Join device DMO	DMO d'appareil de rattachement
Advertisement solicitation	Sollicitation d'annonce
Advertising router DMO	DMO de routeur d'annonce
Advertisement	Annonce
System manager	Gestionnaire de système
Security manager	Gestionnaire de sécurité
Security join request	Demande de rattachement de sécurité
Security join reply	Réponse de rattachement de sécurité
MIC request	Demande de MIC
Security confirm request	Demande de confirmation de sécurité
Security confirm response	Réponse de confirmation de sécurité

Figure 44 – Exemple: Vue d'ensemble du processus de rattachement à clés symétriques



Légende

Anglais	Français
Join device DMO	DMO d'appareil de rattachement
System manager	Gestionnaire de système
Security manager	Gestionnaire de sécurité
Security join request	Demande de rattachement de sécurité
Security join reply	Réponse de rattachement de sécurité
MIC request	Demande de MIC
Security confirm request	Demande de confirmation de sécurité
Security confirm response	Réponse de rattachement de sécurité

Figure 45 – Exemple: Vue d'ensemble du processus de rattachement à clés symétriques d'appareil dorsal

Comme montré à la Figure 44, un nouvel appareil doit utiliser les méthodes définies pour le DMO de routeur d'annonce pour envoyer et recevoir les messages de demande de rattachement et de réponse de rattachement. Les méthodes relatives aux informations autres que de sécurité sont décrites en 6.3.9.2. Les méthodes du DMO relatives aux informations de sécurité sont décrites en 7.4.5.2. Après émission de la DMO.Proxy_Security_Sym_Join().Request, le nouvel appareil doit démarrer le temporisateur de rattachement de comptage JT_1.

Le routeur d'annonce doit utiliser les méthodes définies pour le DMSO du gestionnaire de système pour envoyer et recevoir les messages de demande et de réponse relatifs à la non-sécurité. Ces méthodes de DMSO sont décrites en 6.3.9.5. Les méthodes définies pour l'objet proxy de gestion de sécurité (PSMO) du gestionnaire de système doivent être utilisées par le routeur d'annonce pour envoyer et recevoir les messages de demande et de réponse relatifs à la sécurité. Les méthodes du PSMO relatives aux informations de sécurité sont décrites en 7.4.5.2. Comme montré à la Figure 44, le PSMO reçoit la demande de rattachement relative à

la sécurité et la transmettra au gestionnaire de sécurité. Le gestionnaire de sécurité peut vérifier une liste blanche ou noire pour y détecter l'appareil et demander une vérification humaine avant de décider d'admettre ou rejeter l'appareil se rattachant. Si le résultat est positif, le gestionnaire de sécurité vérifie les informations cryptographiques de la demande de rattachement. Si les vérifications échouent, le gestionnaire de système reçoit l'instruction de révoquer les ressources allouées au nouvel appareil. Si l'essai réussit, le gestionnaire de sécurité agit comme suit:

NOTE 1 La méthodologie du filtrage de l'appareil se rattachant dans le gestionnaire de sécurité ne relève pas du domaine d'application de la présente norme.

a) démarre le temporisateur de rattachement JT_2;

b) génère une nouvelle clé principale pour le nouvel appareil;

c) crée une nouvelle session sécurisée pour le contrat entre le gestionnaire de système et le nouvel appareil;

d) récupère la clé D courante et le CryptoKeyIdentifier pour le sous-réseau du nouvel appareil;

e) génère un nouveau défi unique pour le nouvel appareil;

f) protège cryptographiquement les clés susmentionnées et forme un code de contrôle d'intégrité de message sur la réponse entière; et

g) envoie en retour au PSMO la réponse relative à la sécurité, y compris le code de contrôle d'intégrité de message.

Le PSMO envoie de nouveau cette réponse relative à la sécurité vers le routeur d'annonce qui, à son tour, la transmet au nouvel appareil.

Le nouvel appareil vérifie l'intégrité cryptographique de la réponse APDU relative à la sécurité. Si l'essai échoue, l'APDU reçue est rejetée. Si l'essai réussit, alors la réponse relative à la sécurité est traitée par l'appareil, qui annule le temporisateur de rattachement JT_1.

La réponse de rattachement APDU non relative à la sécurité qui est générée par le DMSO du gestionnaire de système est transmise au gestionnaire de sécurité afin de protéger cryptographiquement les informations dans cette APDU. Une fois que le DMSO reçoit cette APDU protégée, il envoie en retour l'APDU protégée vers le routeur d'annonce qui, à son tour, la transmet au nouvel appareil. Le nouvel appareil vérifie l'intégrité cryptographique de cette APDU protégée avant d'utiliser les informations dans cette APDU pour parachever le processus de rattachement. L'APDU inclut des informations relatives au contrat initial que le gestionnaire de système a établi entre le nouvel appareil et le gestionnaire de système. Ce contrat est décrit en 6.3.11.2.6.7.

Comme partie intégrante de la dernière étape du processus de rattachement, le nouvel appareil doit renvoyer une confirmation de sécurité APDU au gestionnaire de sécurité qui comprend le défi issu du gestionnaire de sécurité, authentifié avec la nouvelle clé principale symétrique partagée. Cette confirmation de sécurité est envoyée au PSMO qui la transmet au gestionnaire de sécurité. La méthode du PSMO utilisé à cet effet est décrite en 7.4.5.2.

Le gestionnaire de sécurité vérifie le message de confirmation. Si l'essai échoue, la réponse de confirmation reçue, l'état de rattachement et les informations placées en cache pour le nouvel appareil doivent être abandonnés. Si l'essai réussit, alors le gestionnaire de sécurité annule le temporisateur JT_2 et envoie en retour une réponse de confirmation vers le nouvel appareil.

Si le nouvel appareil reçoit une réponse positive à la demande de confirmation, il annule alors le temporisateur JT_1.

Le contrat qui a été établi entre le nouvel appareil et le gestionnaire de système au cours du processus de rattachement est utilisé pour prendre en charge ces messages.

NOTE 2 En envoyant la réponse du défi authentifié avec la clé principale, le nouvel appareil prouve au gestionnaire de sécurité qu'il était capable d'extraire la clé principale, et donc qu'il avait la clé de rattachement.

L'ASL peut concaténer des ASDU résultantes d'appels de méthodes multiples vers une seule et même TSDU, garantissant ainsi que si l'une est reçue, toutes sont reçues. Par exemple, pour réduire le surdébit de trafic ou le temps de rattachement, la Proxy_Security_Sym_Join().Request et la Proxy_System_Manager_Contract().Request peuvent être concaténées dans la même TSDU envoyée au DMO du routeur d'annonce.

7.4.5.2 Méthodes d'objet de gestion d'appareil et d'objet proxy de gestion de service relatives au processus de rattachement à clés symétriques

7.4.5.2.1 Généralités

Le nouvel appareil doit utiliser la méthode Proxy_Security_Sym_Join définie pour le DMO du routeur d'annonce dans le routeur d'annonce pour envoyer ses informations de sécurité qui font partie intégrante de la demande de rattachement et pour obtenir ses informations de sécurité qui font partie intégrante de la réponse de rattachement.

NOTE 1 Pour atténuer une inondation de messages de rattachement, le gestionnaire de système limite les ressources sans fil (les intervalles de temps, par exemple) assignées dans des routeurs d'annonce pour recevoir les messages de rattachement. Les ressources sont décrites en 9.3.5.2.4.2.

Le Tableau 59 décrit la méthode Proxy_Security_Sym_Join. L'objet source pour invoquer la DMO.Proxy_Security_Sym_Join().Request doit être le DMO dans le DMAP de l'appareil se rattachant.

Tableau 59 – Méthode Proxy_Security_Sym_Join

Nom du type d'objet normalisé: Device management object (DMO, objet de gestion d'appareil)				
Identificateur du type d'objet normalisé: 127				
Nom de la méthode	**ID de la méthode**	**Description de la méthode**		
Proxy_Security_Sym_Join	5	Méthode d'utilisation du routeur d'annonce en tant que proxy afin d'envoyer des demandes de rattachement relatives à la sécurité et d'obtenir des réponses de rattachement relatives à la sécurité		
	Arguments d'entrée			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	Join_ Request	Security_Sym_Join_Request; voir 7.4.5.2.2	Demande de rattachement relative à la sécurité à clés symétriques à partir d'un nouvel appareil devant être transmise au gestionnaire de sécurité
	Arguments de sortie			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**

Nom du type d'objet normalisé: Device management object (DMO, objet de gestion d'appareil)				
Identificateur du type d'objet normalisé: 127				
Nom de la méthode	**ID de la méthode**	**Description de la méthode**		
	1	Join_ Response	Security_Sym_Join_Response; voir 7.4.5.2.3	Réponse de rattachement relative à la sécurité basée sur les clés symétriques du gestionnaire de sécurité devant être transmise au nouvel appareil. Cela est protégé grâce à la clé de rattachement

Le routeur d'annonce doit utiliser la méthode Security_Sym_Join définie pour le PSMO du gestionnaire de système pour envoyer les informations de sécurité qui font partie intégrante de la demande de rattachement pour le compte du nouvel appareil et pour obtenir les informations de sécurité qui font partie intégrante de la réponse de rattachement.

Le Tableau 60 décrit la méthode Security_Sym_Join.

Tableau 60 – Méthode Security_Sym_Join

Nom du type d'objet normalisé: Proxy security management object (PSMO, objet proxy de gestion de sécurité)				
Identificateur du type d'objet normalisé: 105				
Nom de la méthode	**ID de la méthode**	**Description de la méthode**		
Security_Sym_Join	1	Méthode d'utilisation du PSMO dans le gestionnaire de système pour envoyer des demandes de rattachement relatives à la sécurité et recevoir des réponses de rattachement relatives à la sécurité		
	Arguments d'entrée			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	Join_Request	Security_Sym_Join_Request; voir 7.4.5.2.2	Demande de rattachement relative à la sécurité d'un nouvel appareil au gestionnaire de système
	Arguments de sortie			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	Join_Response	Security_Sym_Join_Response; voir 7.4.5.2.3	Réponse de rattachement relative à la sécurité du gestionnaire de sécurité au nouvel appareil protégé par l'utilisation de la clé de rattachement

Comme partie intégrante de la dernière étape du processus de rattachement, le nouvel appareil doit utiliser la méthode Security_Confirm définie pour le PSMO du gestionnaire de système pour envoyer une confirmation de sécurité au gestionnaire de sécurité.

Le Tableau 61 décrit la méthode Security_Confirm.

Tableau 61 – Méthode Security_Confirm

Nom du type d'objet normalisé: Proxy security management object (PSMO, objet proxy de gestion de sécurité)				
Identificateur du type d'objet normalisé: 105				
Nom de la méthode	**ID de la méthode**	**Description de la méthode**		
Security_Confirm	2	Méthode utilisée par le nouvel appareil pour l'envoi de confirmation de sécurité au gestionnaire de sécurité par l'intermédiaire du PSMO.		
	Arguments d'entrée			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	Security_Sym_Confirm	Security_Sym_Confirm; voir 7.4.5.2.4	Confirmation de sécurité du nouvel appareil au gestionnaire de sécurité
	Arguments de sortie			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	-	-	-	-

NOTE 2 Bien que la méthode Security_Confirm n'ait aucun argument de sortie, le message de réponse Execution dans la sous-couche application est retourné comme résultat de cette méthode.

7.4.5.2.2 Demande de rattachement à clés symétriques

La structure de données Security_Sym_Join_Request qui est utilisée pour former une demande de rattachement à clé symétrique est définie dans le Tableau 62.

Tableau 62 – Structure de données Security_Sym_Join_Request

Nom du type de données normalisé: Security_Sym_Join_Request		
Code du type de données normalisé: 410		
Nom de l'élément	**Identificateur de l'élément**	**Type de l'élément**
New_Device_EUI64	1	Type: EUI64Address Classification: Constant Accessibilité: Read Only
128_Bit_Challenge_From_New_Device	2	Type: SymmetricKey Classification: Static Accessibilité: Read/write
Algorithm_Identifier	3	Type: Unsigned8 Classification: Static Accessibilité: Read Only Valeur par défaut: 1
MIC	4	Type: Unsigned32 Classification: Static Accessibilité: Read Only

Les champs comprennent:

- New_Device_EUI64 est l'adresse EUI64Address de l'appareil de rattachement. Cette adresse EUI64Address est utilisée par le routeur d'annonce lorsqu'il transmet le

message au gestionnaire de système pour identifier cet appareil de façon univoque, car il pourrait y avoir plusieurs nouveaux appareils qui se rattachent en même temps.

- 128_Bit_Challenge_from_new_device est un nouveau défi unique généré par le nouvel appareil pour vérifier que le gestionnaire de sécurité est vivant.
- L'identificateur d'algorithme doit être utilisé pour spécifier l'algorithme à clé symétrique utilisé dans le réseau-cible. La valeur 0x0 doit être réservée. Un algorithme à clé symétrique de 0x01 correspondant à AES_CCM* doit être le seul algorithme symétrique et le seul mode pris en charge pour le processus de rattachement.

NOTE Actuellement, seul AES_CCM* est défini comme étant un algorithme à clé symétrique. Cependant, ce champ est préparé pour des algorithmes pour usage futur ou réglementation nationale.

- Le MIC-32 se calcule sur les éléments 1 à 4, en utilisant la clé de rattachement et les 13 octets de poids fort du défi comme nonce.

7.4.5.2.3 Réponse de rattachement à clés symétriques

La structure de données Security_Sym_Join_Response qui est utilisée pour former une réponse de rattachement à clé symétrique est définie dans le Tableau 63.

Tableau 63 – Structure de données Security_Sym_Join_Response

Nom du type de données normalisé: Security_Sym_Join_Response		
Code du type de données normalisé: 411		
Nom de l'élément	**Identificateur de l'élément**	**Type de l'élément**
128_Bit_Challenge_From_SecurityManager	1	Type: SymmetricKey Classification: Static Accessibilité: Read/write
128_Bit_Response_To_New_Device_Hash_B	2	Type: SymmetricKey Classification: Static Accessibilité: Read/write
Combined_Security_Level	3	Type: Unsigned8 (voir Tableau 64) Classification: Static Accessibilité: Read/write
Master_Key_HardLifeSpan	4	Type: Unsigned16 Classification: Static Accessibilité: Read/write
DL_Key_HardLifeSpan	5	Type: Unsigned16 Classification: Static Accessibilité: Read/write
Sys_Mgr_Session_Key_HardLifeSpan	6	Type: Unsigned16 Classification: Static Accessibilité: Read/write
DL_Key_ID	7	Type: Unsigned8 Classification: Static Accessibilité: Read/write

Nom du type de données normalisé: Security_Sym_Join_Response		
Code du type de données normalisé: 411		
Nom de l'élément	**Identificateur de l'élément**	**Type de l'élément**
Encrypted_DL_Key	8	Type: SymmetricKey Classification: Static Accessibilité: Read/write
Encrypted_Sys_Mgr_Session_Key	9	Type: SymmetricKey Classification: Static Accessibilité: Read/write

Cette structure de données est constituée d'une section en texte clair et d'une section chiffrée. La section en texte clair doit être composée de l'en-tête, du défi d'origine issu du nouvel appareil et d'un nouveau défi issu du gestionnaire de sécurité qui est différent du défi généré par le nouvel appareil, et des politiques principales. La section chiffrée doit être composée de la clé D et de la clé T avec le gestionnaire de système.

- 128_Bit_Challenge_From_Security_Manager est un nouveau défi unique généré par le gestionnaire de sécurité pour vérifier que le nouvel appareil est vivant.
- 128_Bit_Response_To_New_Device_Hash_B doit être calculé comme étant:
  - Hash_B= HMAC-MMO$_{K_join}$[challenge_from_security_manager || challenge_from_new_device || adresse EUI64Address de new_device || adresse EUI64Address de security_manager || Message_Key_Transport]
  - Message_Key_Transport = Combined_Security_Level || Master_Key_HardLifeSpan || DL_Key_HardLifeSpan || Sys_Mgr_Session_Key_HardLifetime || DL_Key_ID || Encrypted D-key || Encrypted SysMan T-key
- Les Master_Key_HardLifeSpan, DL_Key_HardLifeSpan et Sys_Mgr_Session_Key_HardLifeSpan doivent être la HardLifeSpan, exprimée en heures. Les Key Type='001' et Key Usage peuvent être inférés implicitement à partir du Tableau 89 et du Tableau 90 par l'élément Identifier. Une granularité par défaut de 0x2='heures' doit être utilisée pour les politiques dans le message de réponse de rattachement.
- Le DL_Key_ID doit être le CryptoKeyIdentifier associé à la clé D envoyée dans la réponse de rattachement. Le CryptoKeyIdentifier de la clé principale et de la clé T peut être mis implicitement (pas émis, mais inféré) comme étant 0x00.
- Les 13 octets de poids fort du défi envoyé à partir du gestionnaire de sécurité doivent être utilisés comme étant le nonce pour chiffrer la clé-D et la clé-T. La clé-D et la clé-T sont chiffrées en même temps (opération unique de chiffrement AES-CCM* avec une longueur de MIC = 0).
- La réaction à un nouvel appareil doit être le hachage codé défini comme suit:
- La nouvelle clé principale doit être dérivée comme étant:
  - K_master = HMAC-MMOK_join[EUI64Addressnew_device || EUI64Address of security_manager || challenge_from_new_device || challenge_from_security_manager]
- La clé D et la clé T pour prendre en charge le contrat avec le gestionnaire de système doivent être chiffrées en utilisant la nouvelle clé principale.

NOTE 1 En incluant le défi issu du nouvel appareil et en calculant un MIC sur celui-ci, le gestionnaire de sécurité prouve qu'il est un appareil vivant avec connaissance de la clé de rattachement.

NOTE 2 La période de validité de 16 bits, avec les unités en heures, donne une plage de plus de 7 ans, qui est adéquate pour exprimer l'actuelle durée maximale de vie d'une clé.

Tableau 64 – Structure du champ niveau de sécurité compressé

Octet	Bits							
	7	6	5	4	3	2	1	0
1	DL_Security_Level			Sys_Mgr_Ses_Security_Level			Master_key_Sec_level	

Les champs comprennent:

- DL_Security_Level: Le niveau de sécurité appliqué à la clé D acheminée dans le message de réponse Security_Sym_Join. Le format de ce champ doit être spécifié conformément au Tableau 35. Le niveau de sécurité 0, None, et le niveau de sécurité 4, ENC, ne doivent pas être utilisés.
- Sys_Mgr_Ses_Security_Level: Le niveau de sécurité appliqué à la clé T avec le gestionnaire de système acheminée dans le message de réponse Security_Sym_Join. Le format de ce champ doit être spécifié conformément au Tableau 35. Le niveau de sécurité 4, ENC, ne doit pas être utilisé.
- Master_Key_Sec_Level: La taille de MIC appliquée à la clé principale générée dans le processus de rattachement. Le format de ce champ est défini dans le Tableau 65. Sachant que le facteur de chiffrement est différent dans chaque message protégé par la clé principale, seule la taille du MIC est spécifiée dans ce champ. Le niveau de sécurité réel doit être sélectionné dans le Tableau 65 avec une combinaison de conditions de chiffrement dans chaque message.

NOTE 3 Par exemple, sachant que les structures de données Security_New_Session_Request et Security_New_Session_Response n'ont aucun élément devant être chiffré, le Security_Level dans le champ Security_Control est mis à MIC-*n* avec la taille MIC spécifiée dans cette structure. Alors que la structure de données Security_Key_and_Policies a l'élément devant être chiffré, le Security_Level dans le champ Security_Control est mis à ENC-MIC-*n* avec la taille MIC spécifiée dans cette structure.

Tableau 65 – Niveau de sécurité de clé principale

Identificateur de niveau de sécurité	Master_Key_Sec_Level	Attributs de sécurité
0	0	Réservé
1	1	MIC-32
2	2	MIC-64
3	3	MIC-128

NOTE 4 Sachant qu'un MIC est toujours utilisé pour protéger les PDU de processus de rattachement avec la clé principale, l'identificateur de niveau de sécurité 0 est Reserved (réservé).

- ValidNotBefore = temps TAI où l'APDU est reçue;

NOTE 5 La ValidNotBefore peut être inférée comme étant le temps reconstruit utilisé dans l'authentification de la PDU et n'est pas incluse en raison des restrictions d'espace dans la PDU de réponse.

Le temps auquel l'APDU est reçue ne doit pas être supérieur à DSMO.pduMaxAge secondes après qu'elle a été créée. Cela donne un temps de début acceptable.

- HardLifeSpan: La durée valide de la clé est exprimée en heures. La saisie de la valeur 0x0000 doit interdire l'expiration de la clé.

Si la valeur de HardLifeSpan est zéro (effectivement infinie par exemple), alors la durée de vie de la clé inférée doit être:

– ValidNotAfter = 0xFFFF FFFF. Ainsi, cette clé sera perçue par la logique d'expiration de clé comme une clé qui n'expire jamais;

– SoftExpirationTime = ValidNotAfter.

Si la valeur de HardLifeSpan est différente de zéro (finie par exemple), alors la durée de vie de la clé inférée doit être:

– ValidNotAfter = ValidNotBefore + (HardLifeSpan x 3 600);

– SoftExpirationTime = ValidNotBefore + (SoftLifeSpanRatio x HardLifeSpan x 3 600).

Un SoftLifeSpanRatio de 50 % doit être utilisé comme valeur par défaut pour les clés envoyées avec un champ Key_HardLifeSpan.

7.4.5.2.4 Confirmation de la sécurité à clés symétriques

La structure de données Security_Sym_Confirm qui est utilisée pour former une confirmation de sécurité à clé symétrique est définie dans le Tableau 66. L'objet source pour invoquer la PSMO.Proxy_Security_Sym_Confirm().Request doit être le DMO dans le DMAP de l'appareil se rattachant.

Tableau 66 – Structure de données Security_Sym_Confirm

Nom du type de données normalisé: Security_Sym_Confirm		
Code du type de données normalisé: 412		
Nom de l'élément	**Identificateur de l'élément**	**Type de l'élément**
128_Bit_Response_To_Security_Manager	1	Type: SymmetricKey Classification: Static Accessibilité: Read/write

128_Bit_Response_To_Security_Manager doit être calculé comme étant:

HMAC-MMOK_join[challenge_from_new_device || challenge_from_security_manager || EUI64Address of new_device || EUI64Address of security_manager || MIC_1 || MIC_2]

où

MIC_1 est la valeur MIC 32 bits dans la réponse in System_Manager_Join et MIC_2 est la valeur MIC 32 bits dans la réponse System_Manager_Contract.

NOTE 1 La confirmation de rattachement dit au gestionnaire de sécurité que l'appareil était capable de récupérer la clé principale en utilisant la clé de rattachement, fournissant ainsi la preuve qui l'appareil, qui connaît la clé de rattachement, est vivant.

NOTE 2 La construction du hachage pour le protocole de réponse au défi était modélisée d'après le protocole décrit par Menezes et al.:1996, 10.17 (voir Bibliographie).

7.4.6 Processus de rattachement à clés asymétriques

7.4.6.1 Vue d'ensemble

Le processus de rattachement à clés asymétriques, tout comme le processus de rattachement à clés symétriques, spécifie la séquence des étapes par lesquelles et les conditions dans lesquelles un appareil peut devenir une partie du réseau et avoir accès aux informations exigées pour communiquer au sein du réseau, tant avec des appareils immédiatement voisins qu'avec des appareils d'infrastructure particulière, tels que les appareils jouant le rôle de gestionnaire de système ou de gestionnaire de sécurité du réseau. A ce titre, cela nécessite les sous-processus décrits ci-dessous. Noter que la distribution du support de codage et les étapes d'allocation de ressources sont identiques pour les processus de rattachement à clés asymétriques et à clés symétriques. La table des rôles et leur assignation de carte de bits respective sont définies dans l'Annexe B.

Le processus d'inscription comprend:

- Inscription d'adhésion comme membre de réseau. Un appareil et un gestionnaire de sécurité s'engagent dans un protocole mutuel d'authentification d'entité basé sur des techniques à clés asymétriques. Ce protocole fournit la preuve relative à la véritable identité d'appareil tant de l'appareil se joignant que du gestionnaire de sécurité, sur la

base de clés asymétriques authentiques. En outre, l'admission peut être basée sur des critères d'acceptabilité autres que cryptographiques (par exemple, par l'intermédiaire d'un essai d'adhésion comme membre de l'appareil par l'intermédiaire d'une liste de contrôle d'accès). Si l'appareil a été catégoriquement authentifié et est autorisé à rejoindre le réseau, il peut être admis dans le réseau. Le protocole d'authentification d'entité conduit également à l'établissement d'une clé partagée entre l'appareil se joignant et le gestionnaire de sécurité, facilitant ainsi des communications sécurisées et authentiques en cours entre ces appareils.

- Distribution de support de codage. Un gestionnaire de sécurité alloue le support de codage à un appareil nouvellement admis, afin de faciliter des communications ultérieures et l'authentification continue de l'appareil auprès d'autres membres du réseau comme étant un appareil légitime de réseau. Le support de codage peut inclure des clés D, qui sont utilisées pour prouver l'adhésion comme membre de réseau parmi des appareils dans le réseau, et des clés T, qui sont utilisées pour sécuriser et authentifier des communications en cours entre un appareil nouvellement admis et un gestionnaire de système.

Le processus de rattachement suppose que des appareils ont été dotés d'informations suffisantes pour permettre l'authentification correcte de l'appareil. Un appareil se joignant peut tout aussi bien avoir été doté d'informations non relatives à la sécurité.

Le plan d'agrément de clé à clés asymétriques est spécifié en 7.4.6.2, le plan de distribution de clé est spécifié en détail en 7.4.6.3, le plan d'allocation de ressources est spécifié en détail en 6.3.9, et le protocole de rattachement à clés asymétriques est spécifié en 7.4.6.3.5. Les nombres entiers, les octets et les entités utilisés dans le protocole de rattachement à clés asymétriques sont définis dans l'Annexe F. Les blocs modules de base cryptographiques à clés asymétriques sont définis dans l'Annexe H.

7.4.6.2 Plan d'agrément de clé à clés asymétriques

7.4.6.2.1 Vue d'ensemble

7.4.6.2.1.1 Généralités

L'inscription d'adhésion comme membre de réseau est basée sur l'exécution du plan d'agrément de clé à clés asymétriques spécifié en H.4.2 et comporte l'authentification d'appareil basée sur des certificats implicites, tels que spécifiés en H.5.1. Les deux plans impliquent des techniques à clés asymétriques utilisant des courbes elliptiques.

7.4.6.2.1.2 Format de certificat implicite

Le certificat implicite est une preuve d'identité et est utilisé dans le processus de rattachement à clés asymétriques. Il peut acheminer une structure de données arbitraire; cependant, pour permettre l'interfonctionnabilité entre les appareils, le format du certificat implicite, utilisé dans la présente norme, est défini dans le Tableau 67.

Tableau 67 – Format de certificat implicite

Nom de l'élément	Identificateur de l'élément	Type de l'élément
PublicKey_reconstruction_data	1	Type: OctetString37
Sujet	2	Type: EUI64Address
Issuer	3	Type: EUI64Address
Usage_serial_number	4	Type: Structure Usage_Serial (voir Tableau 68)
ValidNotBefore	5	Type: TAITimeRounded
ValidNotAfter	6	Type: TAITimeRounded

- PublicKey_reconstruction_data: Paramètre pour générer une clé publique en utilisant la clé publique de CA.

- Subject: Adresse EUI64Address d'un appareil dont la clé publique/privée est associée à Publickey_Reconstruction_Data
- Issuer: Adresse EUI64Address d'un appareil qui a généré ce certificat.
- Usage_serial_number: Indiquant un usage de certification et un numéro de série.
- ValidNotBefore: Temps TAI absolu (en secondes) où ce certificat devient valide.
- ValidNotAfter: Temps TAI absolu (en secondes) où ce certificat devient non valide.

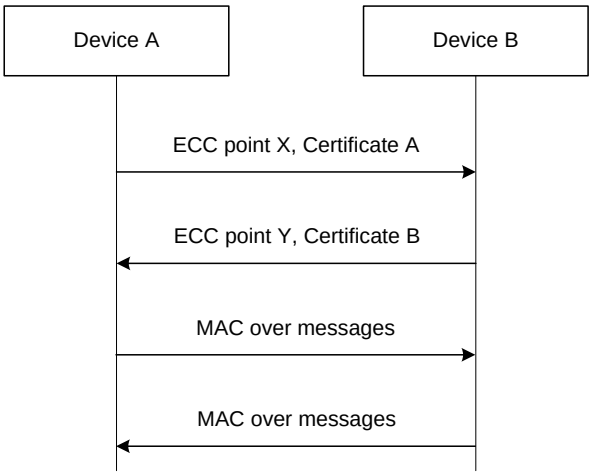
Tableau 68 – Structure d'Usage_serial_number

Octet	Bits							
	7	6	5	4	3	2	1	0
1	Reserved	Issuable	Serial_number					

- Reserved: Il convient que le champ Reserved soit 0.
- Issuable: Si ce champ est 0, la paire de clés correspondant à ce certificat ne doit pas être utilisée pour signer un autre certificat. Autrement, la paire de clé peut être utilisée pour signer un autre certificat.
- Serial_number: Numéro de série de ce certificat géré par l'émetteur.

7.4.6.2.2 Description du plan

La Figure 46 montre la messagerie impliquée dans le plan d'agrément de clé à clés asymétriques utilisé avec la présente norme.



Légende

Anglais	Français
Device A	Appareil A
Device B	Appareil B
ECC point X, certificate A	Point X sur ECC, certificat A
ECC point Y, certificate B	Point Y sur ECC, certificat B
MAC over messages	Messages au-dessus de MAC

Figure 46 – Plan d'agrément de clé à clés asymétriques

Dans le contexte du protocole de rattachement, le plan d'agrément de clé implique une messagerie entre un appareil se joignant et un gestionnaire de sécurité, par laquelle l'appareil se joignant lance le protocole et par laquelle le gestionnaire de sécurité agit comme ledit répondeur. Donc, dans les termes de la Figure 46, l'appareil se joignant joue le rôle d'appareil A et le gestionnaire de sécurité joue le rôle d'appareil B.

Le protocole inclut les composantes séquentielles suivantes:

a) Contributions de clé. Chaque partie prenante génère au hasard une paire de clés publiques de court terme (éphémères) et communique la clé publique éphémère (mais pas la clé privée) à l'autre partie prenante. En outre, chaque partie prenante communique le certificat de sa clé publique (statique) à long terme à l'autre partie prenante.

b) Etablissement de clé. Chaque partie prenante calcule la clé partagée en se basant sur les points de courbes elliptiques statiques et éphémères qu'elle a reçus en provenance de l'autre partie prenante, et également en se basant sur les clés privées statiques et éphémères qu'elle génère elle-même. En raison des propriétés des courbes elliptiques, l'une ou l'autre partie prenante arrive à la même clé partagée.

c) Authentification de clé. Chaque partie prenante vérifie l'authenticité de la clé statique à long terme de l'autre partie prenante, pour obtenir la preuve que la seule partie prenante qui peut être capable de calculer la clé partagée est effectivement la partie prenante communicante perçue.

d) Confirmation de clé. Chaque partie prenante calcule et communique une valeur de contrôle d'authentification de message sur des chaînes communiquées par l'autre partie prenante, pour prouver la possession de la clé partagée à l'autre partie prenante. Cela confirme à chaque partie prenante la véritable identité de l'autre partie prenante et prouve que l'autre partie prenante a calculé avec succès la clé partagée. Ce message de confirmation de clé peut authentifier une chaîne complémentaire communiquée par la partie prenante elle-même. Les chaînes et les opérations sur les chaînes sont définies dans l'Annexe F.

Le protocole suppose que chaque partie prenante a accès à la clé racine de l'autorité de certification (CA[7]) qui a signé le certificat reçu en provenance de l'autre partie prenante.

7.4.6.2.3 Propriétés de sécurité du plan

L'exécution réussie du plan complet conduit à des propriétés de sécurité, y compris ce qui suit:

- Authentification mutuelle d'entité. Chaque partie prenante a des assurances quant à la véritable identité de l'autre partie prenante et quant au fait que cette partie prenante était vivante pendant l'exécution du protocole.
- Authentification de clé implicite mutuelle. Chaque partie prenante a des assurances que la seule partie prenante qui a pu avoir été capable de calculer la clé partagée est effectivement la partie prenante communicante prévue.
- Confirmation de clé mutuelle. Chaque partie prenante a la preuve que sa partie prenante communicante prévue a calculé avec succès la clé partagée.
- Secret direct parfait. La compromission de la clé statique ne compromet pas les clés partagées passées.
- Aucune commande de clé unilatérale. Chaque partie prenante a l'assurance qu'aucune partie prenante n'était capable de commander ou prédire la valeur de la clé partagée.
- Propriétés de sécurité complémentaires, telles que la résilience de partage de clé inconnue et sécurité de clé connue. Pour des détails, voir la norme ANSI X9.63:2011, Tableau H.2.

Les services de sécurité fournis par chaque plan sont assurés après l'achèvement réussi du plan complet en question (et si les conditions préalables du plan sont respectées). A partir des plans, il n'est pas a priori clair quelles propriétés sont fournies pendant l'exécution des étapes de protocole du plan. Les services de sécurité fournis incluent:

– Le traitement des contributions de clé aléatoires n'offre aucun service de sécurité, car ces messages sont indépendants.

_____

[7] CA = *certificate authority*.

– A partir de la perspective de l'appareil se rattachant A, le protocole est fini après l'achèvement des étapes de traitement résultant de la réception du message de confirmation de clé MAC_B, alors qu'à partir de la perspective du gestionnaire de sécurité B, le protocole est seulement fini après l'achèvement des étapes de traitement résultant de la réception du message de confirmation de clé MAC_A. En particulier, le gestionnaire de sécurité B n'a aucune assurance avant la réception et le traitement du message de confirmation clé MAC_A. Ainsi, toutes les éventuelles actions par B déclenchées avant l'achèvement du protocole entier avec A sont prématurées, en ce sens qu'elles ne peuvent pas être logiquement basées sur d'éventuelles assurances de sécurité (car il n'y en a aucune). En revanche, toutes les éventuelles actions par B déclenchées après que l'achèvement réussi du protocole entier avec A peuvent être bien fondées, en ce sens qu'elles peuvent être basées sur les services de sécurité résultant de l'achèvement du protocole.

NOTE Cela souligne de nouveau l'importance de la prise en considération de l'effet des plans cryptographiques dans leur intégralité.

7.4.6.3 Plan de distribution de clé

7.4.6.3.1 Vue d'ensemble

La distribution de clé est basée sur la clé partagée résultant du plan d'agrément de clé à clés asymétriques, tel que décrit en 7.4.6.2, exécuté entre l'appareil se joignant et le gestionnaire de sécurité.

7.4.6.3.2 Description du plan

Le mécanisme pour la distribution du support de codage du gestionnaire de sécurité vers l'appareil nouvellement rattaché et le gestionnaire de système est le même que celui décrit dans le processus de rattachement à clés symétriques. Pour les détails, voir 7.4.4.

7.4.6.3.3 Propriétés de sécurité du plan

L'exécution réussie du plan de distribution de clé conduit à des propriétés de sécurité, y compris ce qui suit:

- Transfert sécurisé et authentique de la clé D et des informations de codage associées du gestionnaire de sécurité vers l'appareil nouvellement rattaché.
- Transfert sécurisé et authentique de la clé T et des informations de codage associées du gestionnaire de sécurité vers l'appareil nouvellement rattaché et vers le gestionnaire de système sélectionné par le gestionnaire de sécurité.
- Dans un cas comme dans l'autre, le support de codage distribué est généré par le gestionnaire de sécurité, offrant donc la commande de clé unilatérale.

7.4.6.3.4 Formats de messagerie de protocole

Le mécanisme pour la distribution du support de codage du gestionnaire de sécurité vers l'appareil nouvellement rattaché et le gestionnaire de système est le même que celui décrit dans le processus de rattachement à clés symétriques. Pour les détails, voir 7.4.4.

7.4.6.3.5 Protocole de rattachement à clés asymétriques

Le protocole de rattachement à clés asymétriques peut être vu comme un protocole qui combine le plan d'agrément de clé à clés asymétriques débattu en 7.4.6.2 et le plan de distribution de clé débattu en 7.4.6.3, la différence principale résidant dans l'organisation réelle de la messagerie en trames.

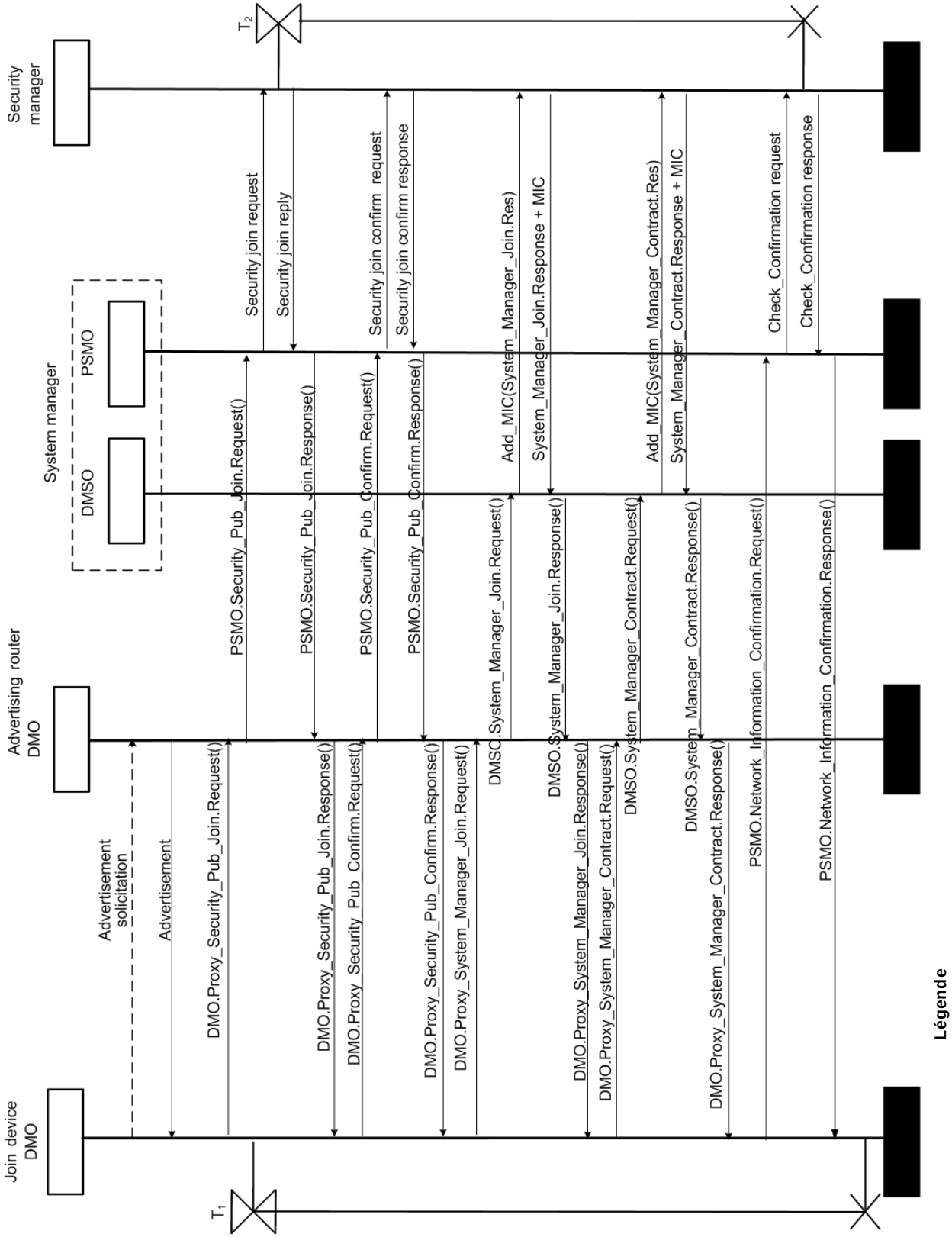
Le protocole de rattachement à clés asymétriques et le protocole de rattachement à clés symétriques diffèrent seulement dans l'utilisation d'un plan d'agrément de clé à clés asymétriques, plutôt que d'un plan d'agrément de clé à clés symétriques. Ainsi, tous autres

aspects de la spécification du protocole de rattachement à clés symétriques (voir 7.4.4) s'appliquent également au protocole de rattachement à clés asymétriques.

7.4.6.4 Messages de processus de rattachement à clés asymétriques

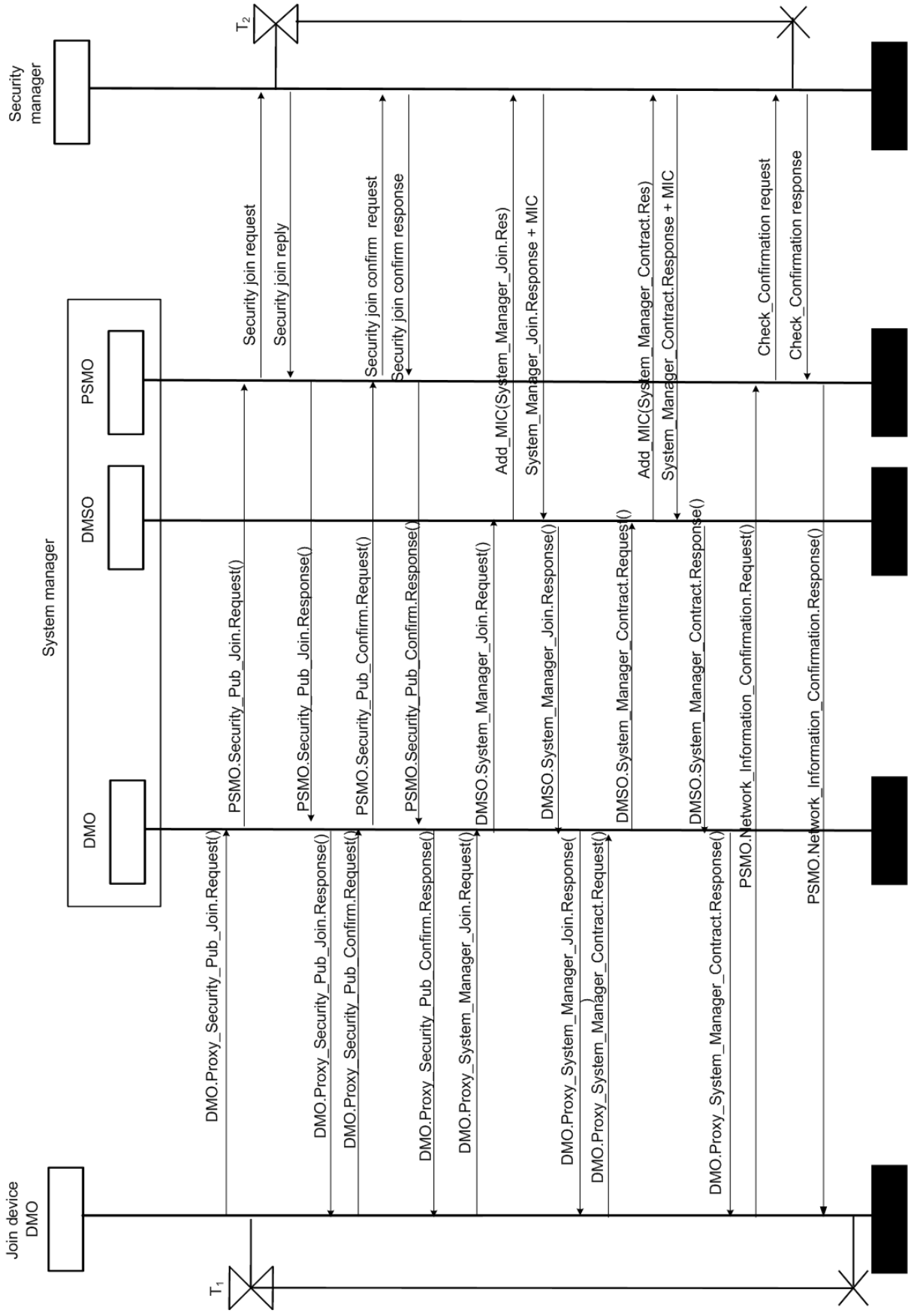
7.4.6.4.1 Généralités

La Figure 47 et la Figure 48 illustrent la messagerie impliquée dans le processus de rattachement à clés asymétriques par lequel un nouvel appareil doit rejoindre un réseau en fonctionnement dans lequel il n'a pas été récemment un participant. Le flux montre le cas normal dans lequel il ne se produit pas d'erreur ou de temporisation. Les temporisations sont spécifiées en 7.4.7.3.



Anglais	Français
Join device DMO	DMO appareil de rattachement
Advertisement solicitation	Sollicitation d'annonce
Advertising router DMO	DMO de routeur d'annonce
Advertisement	Annonce
System manager	Gestionnaire de système
Security manager	Gestionnaire de sécurité
Security join request	Demande de rattachement de sécurité
Security join reply	Réponse de rattachement de sécurité
Security join confirm request	Demande de confirmation de rattachement de sécurité
Security join confirm response	Réponse de confirmation de rattachement de sécurité

Figure 47 – Exemple: Vue d'ensemble du processus de rattachement à clés asymétriques pour un appareil avec une DL



Anglais	Français
Join device DMO	DMO appareil de rattachement
System manager	Gestionnaire de système
Security manager	Gestionnaire de sécurité
Security join request	Demande de rattachement de sécurité
Security join reply	Réponse de rattachement de sécurité
Security join confirm request	Demande de confirmation de rattachement de sécurité
Security join confirm response	Réponse de confirmation de rattachement de sécurité

Figure 48 – Exemple: Vue d'ensemble du processus de rattachement à clés asymétriques d'un appareil dorsal

Sur l'appareil se rattachant, le processus de rattachement à clés asymétriques doit être initié par l'émission d'une DMO.Proxy_Security_Pub_Join().Request et finalisé par la réception d'une PSMO.Network_Information_Confirmation().Response valide. Sur le gestionnaire de sécurité, le processus de rattachement à clés asymétriques doit être initié par la réception d'un message valide dérivé à partir d'une PSMO.Security_Pub_Join().Request et finalisé par un message d'émission dérivé pour être une PSMO.Network_Information_Confirmation().Response.

Comme montré à la Figure 48, un nouvel appareil doit utiliser les méthodes définies pour le DMO de routeur d'annonce pour envoyer et recevoir les messages de demande de rattachement et de réponse de rattachement. Les méthodes relatives aux informations autres que de sécurité sont décrites en 6.3.9.2. Les méthodes de DMO relatives aux informations de sécurité pour la méthode de rattachement asymétrique sont décrites en 7.4.6.4.2.

Le routeur d'annonce doit utiliser les méthodes définies pour le DMSO du gestionnaire de système pour envoyer et recevoir les messages de demande et de réponse relatifs à la non-sécurité. Ces méthodes de DMSO sont décrites en 6.3.9.5. Les méthodes définies pour l'objet proxy de gestion de sécurité (PSMO) du gestionnaire de système doivent être utilisées par le routeur d'annonce pour envoyer et recevoir les messages de demande et de réponse relatifs à la sécurité. Les méthodes du PSMO relatives aux informations de sécurité sont décrites en 7.4.5.2.

7.4.6.4.2 Méthodes d'objet de gestion d'appareil et d'objet proxy de gestion de sécurité relatives au processus de rattachement à clés asymétriques

Le nouvel appareil doit utiliser la méthode Proxy_Security_Pub_Join définie pour le DMO de routeur d'annonce dans le routeur d'annonce pour envoyer ses informations de sécurité qui font partie intégrante de la demande de rattachement et pour obtenir ses informations de sécurité qui font partie intégrante de la réponse de rattachement. Après émission de la DMO.Proxy_Security_Pub_Join().Request, le nouvel appareil doit démarrer le temporisateur de rattachement JT_1. Après réception de la DMO.Proxy_Security_Pub_Join().Request, le gestionnaire de sécurité doit démarrer le temporisateur de rattachement JT_2.

Le Tableau 69 décrit la méthode Proxy_Security_Pub_Join.

Tableau 69 – Méthode Proxy_Security_Pub_Join

<table>
<tr><th colspan="5">Nom du type d'objet normalisé: Device management object (DMO, objet de gestion d'appareil)</th></tr>
<tr><th colspan="5">Identificateur du type d'objet normalisé: 127</th></tr>
<tr><th>Nom de la méthode</th><th>ID de la méthode</th><th colspan="3">Description de la méthode</th></tr>
<tr><td rowspan="8">Proxy_Security_Pub_Join</td><td>6</td><td colspan="3">Méthode d'utilisation du routeur d'annonce en tant que proxy afin d'envoyer des demandes de rattachement relatives à la sécurité et d'obtenir des réponses de rattachement relatives à la sécurité</td></tr>
<tr><td colspan="4">Arguments d'entrée</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td>1</td><td>Join_Request</td><td>Security_Pub_Join_Request; voir 7.4.6.4.3</td><td>Demande de rattachement relative à la sécurité basée sur les clés publiques à partir d'un nouvel appareil devant être transmise au gestionnaire de sécurité</td></tr>
<tr><td colspan="4">Arguments de sortie</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td>1</td><td>Join_Response</td><td>Security_Pub_Join_Response; voir 7.4.6.4.3</td><td>Réponse de rattachement relative à la sécurité basée sur les clés publiques du gestionnaire de sécurité devant être transmise au nouvel appareil. Cela est protégé grâce à la clé de rattachement</td></tr>
</table>

Le routeur d'annonce doit utiliser la méthode Security_Pub_Join définie pour le PSMO du gestionnaire de système pour envoyer les informations de sécurité qui font partie intégrante de la demande de rattachement pour le compte du nouvel appareil et pour obtenir les informations de sécurité qui font partie intégrante de la réponse de rattachement.

L'objet source de la DMO.Proxy_Security_Pub_Join().Request doit être le DMO dans le DMAP de l'appareil se rattachant.

Le Tableau 70 décrit la méthode Security_Pub_Join.

Tableau 70 – Méthode Security_Pub_Join

<table>
<tr><th colspan="5">Nom du type d'objet normalisé: Proxy security management object (PSMO, objet proxy de gestion de sécurité)</th></tr>
<tr><th colspan="5">Identificateur du type d'objet normalisé: 105</th></tr>
<tr><th>Nom de la méthode</th><th>ID de la méthode</th><th colspan="3">Description de la méthode</th></tr>
<tr><td rowspan="6">Security_Pub_Join</td><td>3</td><td colspan="3">Méthode d'utilisation du PSMO dans le gestionnaire de système pour envoyer des demandes de rattachement relatives à la sécurité et recevoir des réponses de rattachement relatives à la sécurité</td></tr>
<tr><td colspan="4">Arguments d'entrée</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td>1</td><td>Join_Request</td><td>Security_Pub_Join _Request; voir 7.4.6.4.3</td><td>Demande de rattachement relative à la sécurité d'un nouvel appareil au gestionnaire de système</td></tr>
<tr><td colspan="4">Arguments de sortie</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td>1</td><td>Join_Response</td><td>Security_Pub_Join _Response; voir 7.4.6.4.3</td><td>Réponse de rattachement relative à la sécurité du gestionnaire de sécurité au nouvel appareil protégé par l'utilisation de la clé de rattachement</td></tr>
</table>

Après avoir reçu le message Proxy_Security_Pub_Join().Response, le nouvel appareil doit utiliser la méthode Proxy_Security_Pub_Confirm() définie pour le DMO de routeur d'annonce pour envoyer une confirmation de sécurité au routeur d'annonce. Le Tableau 71 décrit cette méthode. L'objet source de la DMO.Proxy_Security_Pub_Join().Request doit être le DMO dans le DMAP de l'appareil se rattachant.

Le routeur d'annonce doit utiliser la méthode Security_Pub_Confirm définie pour le PSMO du gestionnaire de système pour envoyer ces informations de sécurité au gestionnaire de sécurité. Le Tableau 72 décrit cette méthode.

Le gestionnaire de sécurité est responsable de vérifier le message de confirmation. Si l'essai échoue, l'état de rattachement et les informations placées en cache pour le nouvel appareil doivent être initialisés ou abandonnés. Si l'essai réussit, alors le gestionnaire de sécurité arrête le temporisateur de rattachement JT_2 et envoie une réponse de confirmation et les réponses d'informations autres que de sécurité au nouvel appareil.

Si le nouvel appareil reçoit une réponse valide à sa demande de confirmation, alors le nouvel appareil arrête le temporisateur JT_1.

Tableau 71 – Méthode Proxy_Security_Pub_Confirm

Nom du type d'objet normalisé: Device management object (DMO, objet de gestion d'appareil)				
Identificateur du type d'objet normalisé: 127				
Nom de la méthode	**ID de la méthode**	**Description de la méthode**		
Proxy_Security_Pub_Confirm	7	Méthode d'utilisation du routeur d'annonce en tant que proxy pour l'envoi de confirmation de sécurité		
	Arguments d'entrée			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	Security_Pub_Confirm_Request	Security_Pub_Confirm_Request; voir 7.4.6.4.3	Confirmation de sécurité du nouvel appareil au gestionnaire de sécurité par l'intermédiaire du routeur d'annonce
	Arguments de sortie			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	Security_Pub_Confirm_Response	Security_Pub_Confirm_Response; voir 7.4.6.4.3	Confirmation de sécurité du gestionnaire de sécurité au nouvel appareil par l'intermédiaire du routeur d'annonce

Tableau 72 – Méthode Security_Pub_Confirm

<table>
<tr><th colspan="5">Nom du type d'objet normalisé: Proxy security management object (PSMO, objet proxy de gestion de sécurité)</th></tr>
<tr><th colspan="5">Identificateur du type d'objet normalisé: 105</th></tr>
<tr><th>Nom de la méthode</th><th>ID de la méthode</th><th colspan="3">Description de la méthode</th></tr>
<tr><td rowspan="6">Security_Pub_Confirm</td><td>4</td><td colspan="3">Méthode utilisée pour l'envoi de confirmation de sécurité du nouvel appareil au gestionnaire de sécurité par l'intermédiaire du PSMO.</td></tr>
<tr><td colspan="4">Arguments d'entrée</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td>1</td><td>Security_Pub_Confirm_Request</td><td>Security_Pub_Confirm_Request; voir 7.4.6.4.3</td><td>Confirmation de sécurité du nouvel appareil au gestionnaire de sécurité par l'intermédiaire du routeur d'annonce</td></tr>
<tr><td colspan="4">Arguments de sortie</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td>1</td><td>Security_Pub_Confirm_Response</td><td>Security_Pub_Confirm_Response; voir 7.4.6.4.3</td><td>Confirmation de sécurité du gestionnaire de sécurité au nouvel appareil par l'intermédiaire du routeur d'annonce</td></tr>
</table>

Après avoir reçu la réponse DMO.Proxy_System_Manager_Join() et DMO.Proxy_System_Manager_Contract(), l'appareil de rattachement invoque la méthode PSMO.Network_Information_Confirmation(). Le Tableau 73 décrit la méthode.

Le champ Confirm dans la PSMO.Network_Information_Confirmation().Request est le MACTag généré avec l'opération suivante:

$$MACTag = HMAC\text{-}MMOK\_join[MIC_1 \,||\, MIC_2 \,||\, Challenge_{joining\_device}]$$

où:

MIC_1: Champ MIC dans la DMO.Proxy_System_Manager_Join().Response (voir Tableau 19);

MIC_2: Champ MIC dans la DMO.Proxy_System_Manager_Contract().Response (voir Tableau 20).

Tableau 73 – Méthode Network_Information_Confirmation

<table>
<tr><th colspan="5">Nom du type d'objet normalisé: Proxy security manager object (PSMO, objet proxy de gestion de sécurité)</th></tr>
<tr><th colspan="5">Identificateur du type d'objet normalisé: 105</th></tr>
<tr><th>Nom de la méthode</th><th>ID de la méthode</th><th colspan="3">Description de la méthode</th></tr>
<tr><td rowspan="6">Network_Information_Confirmation</td><td>5</td><td colspan="3">Méthode utilisée pour s'assurer que l'information réseau correcte a été reçue par l'appareil de rattachement</td></tr>
<tr><td colspan="4">Arguments d'entrée</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td>1</td><td>Confirmation</td><td>OctetString16</td><td>Message de confirmation assurant que l'appareil de rattachement a reçu l'information réseau correcte du gestionnaire de système</td></tr>
<tr><td colspan="4">Arguments de sortie</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td>-</td><td>-</td><td>-</td><td>-</td></tr>
</table>

7.4.6.4.3 Formats de messagerie de protocole

7.4.6.4.3.1 Format de la structure interne de la demande de rattachement (PK-join-1)

Le type de données Security_Pub_Join_Request utilisé dans la méthode de Security_Pub_Join et dans la méthode Proxy_Security_Pub_Join a la structure suivante et représente le premier flux de message du plan d'agrément de clé à clés asymétriques (7.4.6.2). Ce type de données est utilisé par le nouvel appareil et son routeur proxy dans les méthodes correspondantes du DMO du routeur proxy et du PSMO du gestionnaire de système respectivement. Les données PK-join-1 doivent être formatées telles que montrées dans le Tableau 74.

Tableau 74 – Format de la structure interne de la demande de rattachement asymétrique

Nom du type de données normalisé: Security_Pub_Join_Request (PK-Join-1)		
Code du type de données normalisé: 415		
Nom de l'élément	**Identificateur de l'élément**	**Type de l'élément**
New_Device_EUI64	1	Type: EUI64Address Classification: Constant Accessibilité: Read Only
Champ de contrôle de protocole	2	Type: Unsigned8 Classification: Constant Accessibilité: Read Only Valeur par défaut: 1000 0000
Point X de courbe elliptique éphémère	3	Type: OctetString37 Classification: Static Accessibilité: Read Only
Certificat implicite du nouvel appareil	4	Type: OctetString SIZE(37..66) Classification: Static Accessibilité: Read/write
NOTE 1 Le format du certificat implicite utilisé dans la présente norme est défini en 7.4.6.2.1.2. NOTE 2 La taille totale de la demande de rattachement à clés asymétriques se situe dans la plage de 83..112 octets. Si l'utilisateur emploie cette approche, la demande exigera parfois plus d'une DPDU d'acheminement.		

Le champ commande de protocole a une taille de 1 octet et spécifie l'algorithme qui est utilisé pour le protocole de rattachement à clés asymétriques et l'étape du protocole qui est actuellement exécutée. Ce sous-champ doit être formaté comme spécifié dans le Tableau 75.

Tableau 75 – Format du champ commande de protocole

7	6	5	4	3	2	1	0
ID de l'algorithme = "10"		Réservé				Phase du sous-protocole de rattachement	

L'ID de l'algorithme a une taille de 2 bits et indique l'algorithme de rattachement à clé asymétrique en cours d'utilisation. L'algorithme défini dans la présente norme a pour ID 0b'10'.

La phase du sous-protocole de rattachement a une taille de 2 bits et indique la phase en cours du protocole:

0: Demande de rattachement à clés asymétriques;

1: Réponse de rattachement à clés asymétriques;

2: Demande de confirmation de rattachement à clés asymétriques;

3: Réponse de confirmation de rattachement à clés asymétriques.

7.4.6.4.3.2 Format de la structure interne de la réponse de rattachement asymétrique (PK-join-2)

Le type de données Security_Pub_Join_Response utilisé dans la méthode de Security_Pub_Join et dans la méthode Proxy_Security_Pub_Join a la structure suivante et représente le premier flux de message du plan d'agrément de clé à clés asymétriques

(7.4.6.2). Les données PK-join-2 doivent être formatées telles que montrées dans le Tableau 76.

Tableau 76 – Format de la structure interne de la réponse de rattachement asymétrique

Nom du type de données normalisé: Security_Pub_Join_Response (PK-Join-2)		
Code du type de données normalisé: 416		
Nom de l'élément	**Identificateur de l'élément**	**Type de l'élément**
New_Device_EUI64	1	Type: EUI64Address Classification: Constant Accessibilité: Read Only
Champ de contrôle de protocole	2	Type: Unsigned8 Classification: Constant Accessibilité: Read Only Valeur par défaut: 1000 0001
Point Y de courbe elliptique éphémère	3	Type: OctetString37 Classification: Static Accessibilité: Read Only
Certificat implicite du gestionnaire de sécurité	4	Type: OctetString SIZE(37..66) Classification: Static Accessibilité: Read/write
NOTE 1 Le format du certificat implicite utilisé dans la présente norme est défini en 7.4.6.2.1.2. NOTE 2 La taille totale de la demande de rattachement à clés asymétriques se situe dans la plage de 83..112 octets. Si l'utilisateur emploie cette approche, la demande exigera parfois plus d'une DPDU d'acheminement.		

7.4.6.4.3.3 Format de la première structure interne de la confirmation de rattachement (PK-join-3)

Le type de données Security_Pub_Confirm_Request utilisé dans la méthode de Security_Pub_Confirm et dans la méthode Proxy_Security_Pub_Confirm a la structure suivante et représente le troisième flux de message du plan d'agrément de clé à clés asymétriques (7.4.6.2). Les données PK-join-3 doivent être formatées telles que montrées dans le Tableau 77.

Tableau 77 – Format de la première structure interne de la confirmation de rattachement

Nom du type de données normalisé: Security_Pub_Confirm_Request (PK-Join-3)		
Code du type de données normalisé: 417		
Nom de l'élément	**Identificateur de l'élément**	**Type de l'élément**
New_Device_EUI64	1	Type: EUI64Address Classification: Constant Accessibilité: Read Only
Champ de contrôle de protocole	2	Type: Unsigned8 Classification: Constant Accessibilité: Read Only Valeur par défaut: 1000 0010
Message_authentication_tag_MAC	3	Type: OctetString16 Classification: Static Accessibilité: Read Only
Taille du texte	4	Type: Unsigned8 Classification: Static Accessibilité: Read/write Valeur par défaut: 0 Plage valide: 0..31
Texte	5	Type: OctetStringN (SIZE: voir élément 4) Classification: Static Accessibilité: Read/write

Le Message_authentication_tag_MAC est généré par la formule suivante.

$$\text{Message\_authentication\_tag\_MAC} = \text{MACmackey}(02_{16} \,||\, U \,||\, V \,||\, QEU \,||\, QEV)$$

où:

U est l'adresse EUI64Address du nouvel appareil;

V est l'ID de 8 octets du gestionnaire de sécurité;

QEU est la chaîne d'octets de la clé publique éphémère du nouvel appareil;

QUV est la chaîne d'octets de la clé publique éphémère du gestionnaire de sécurité.

Cela fait partie du plan d'agrément de clé ECMQV. La MACmackey est définie dans l'ANSI X9.63:2011, 5.7. Dans la présente spécification, la fonction de hachage codé HMAC-MMO avec la clé principale doit être utilisée pour la fonction MACmackey.

Le champ texte est utilisé pour stocker n'importe quelle information qui a besoin d'être authentifiée. Les utilisateurs peuvent utiliser ce champ pour n'importe quelle information qui requiert une protection au cours du processus de rattachement à clés asymétriques.

7.4.6.4.3.4 Format de la seconde structure interne de confirmation de rattachement

Le type de données Security_Pub_Confirm_Response utilisé dans la méthode de Security_Pub_Confirm et dans la méthode Proxy_Security_Pub_Confirm a la structure de données suivante et représente le quatrième flux de message du plan d'agrément de clé à clés asymétriques (7.4.6.2). Les données PK-join-4 doivent être formatées telles que montrées dans le Tableau 78.

Tableau 78 – Format de la structure interne de la réponse de confirmation de rattachement

Nom du type de données normalisé: Security_Pub_Confirm_Response (PK-Join-4)		
Code du type de données normalisé: 418		
Nom de l'élément	**Identificateur de l'élément**	**Type de l'élément**
Champ de contrôle de protocole	1	Type: Unsigned8 Classification: Constant Accessibilité: Read Only Valeur par défaut: 1000 0011
Message_authentication_tag_MAC	2	Type: OctetString16 Classification: Static Accessibilité: Read Only
Taille du texte	3	Type: Unsigned8 Classification: Static Accessibilité: Read/write Valeur par défaut: 0
Texte	4	Type: OctetString (SIZE: voir élément 3) Classification: Static Accessibilité: Read/write
Master_Key_HardLifeSpan	5	Type: Unsigned16 Classification: Static Accessibilité: Read/write
Encrypted D-key (clé D chiffrée)	6	Type: SymmetricKey Classification: Static Accessibilité: Read/write
DL CryptoKeyIdentifier (Identificateur de clé crypto DL)	7	Type: Unsigned8 Classification: Static Accessibilité: Read/write
DL_Key_HardLifeSpan	8	Type: Unsigned16 Classification: Static Accessibilité: Read/write
Encrypted system manager T-key (clé T chiffrée de gestionnaire de système)	9	Type: SymmetricKey Classification: Static Accessibilité: Read/write
System_Manager_Session_Key_HardLifeSpan	10	Type: Unsigned16 Classification: Static Accessibilité: Read/write

Le Message_authentication_tag_MAC est généré par la formule suivante:

$$\text{Message\_authentication\_tag\_MAC} = \text{MACmackey}(03_{16} \,||\, U \,||\, V \,||\, QEU \,||\, QEV)$$

où:

U est l'adresse EUI64Address du nouvel appareil;
V est l'ID de 8 octets du gestionnaire de sécurité;

QEU est la chaîne d'octets de la clé publique éphémère du nouvel appareil;

QUV est la chaîne d'octets de la clé publique éphémère du gestionnaire de sécurité.

Cela fait partie du plan d'agrément de clé ECMQV. La MACmackey est définie dans l'ANSI X9.63:2011, 5.7. Dans la présente spécification, HMAC-MMO avec la clé principale doit être utilisée pour la fonction MACmackey.

Le champ texte est utilisé pour stocker les informations déterminées par un utilisateur qui ont besoin d'être authentifiées. Les utilisateurs peuvent utiliser ce champ pour n'importe quelle information devant être protégée au cours du processus de rattachement à clés asymétriques.

Les champs "key material" (support de clé) et "policy" (politique) sont les mêmes que pour le processus de rattachement à clés symétriques. Spécifiquement, les suivants doivent être les mêmes qu'en 7.4.5:

- Master Key compressed policy (politique compressée de clé principale)
- Encrypted D-key (clé D chiffrée)
- DL CryptoKeyIdentifier (Identificateur de clé crypto DL)
- D-key compressed policy (politique compressée de clé D)
- Encrypted system manager T-key (clé T chiffrée de gestionnaire de système)
- System manager T-key compressed policy (politique compressée de clé T de gestionnaire de système)

7.4.7 Rétablissement après défaillance de processus de rattachement et de durée de vie d'appareil

7.4.7.1 Généralités

A un point quelconque au cours du processus de rattachement, il y a une possibilité qu'une PDU soit abandonnée. Dans ce cas, il convient que le système soit capable de récupérer et continuer. La définition d'état et la transition d'état suivantes décrivent les grandes lignes du mécanisme de rétablissement, avec les effets secondaires déclenchés.

7.4.7.2 Etats d'appareil pendant le processus de rattachement et la durée de vie d'appareil

Les états de l'appareil au cours du processus de rattachement sont:

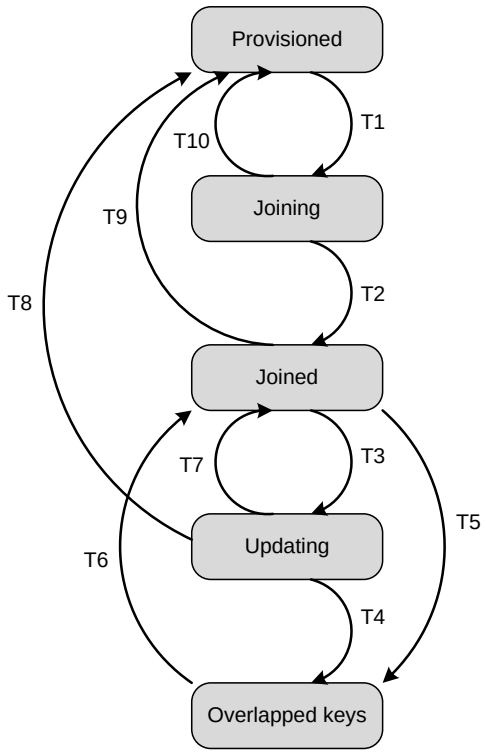
- Provisioned (configuré): Aucune clé principale, n'est pas en train d'obtenir une clé principale.
- Joining (se rattachant): Aucune clé principale, est en train d'obtenir une clé principale.
- Joined (rattaché): Ayant une clé principale courante, n'est pas en train d'obtenir la clé principale suivante.
- Updating (se mettant à jour): Ayant une clé principale courante, est en train d'obtenir la clé principale suivante.
- Overlapped (superposé): Ayant à la fois la clé principale courante et la clé principale suivante.

7.4.7.3 Transitions d´état

Les transitions d'état pour un appareil rejoignant le réseau doivent être telles que présentées dans le Tableau 79 et dans la Figure 49. Les valeurs de temporisation pour le processus de rattachement join_timeout (ID 4) dans le Tableau 92, doivent être configurables en utilisant des annonces de DL. Il convient que l'effacement de clés soit au moins l'équivalent du nettoyage de données confidentielles, conformément à la NIST SP 800-88:2012, Tableau 2-1.

Tableau 79 – Diagramme d'états du processus de rattachement et de la durée de vie d'appareil

Transition	Etat actuel	Evénement(s)	Action(s)	Etat suivant
T1	Provisioned (configuré)	DMO initialise le processus de rattachement	Routeur d'annonce DMO.Proxy_Security_Sym_ Join().Request or DMO.Proxy_Security_Pub_ Join().Request	Joining (se rattachant)
T2	Joining (se rattachant)	DMO.Proxy_Security_Sym_Join ().Response or DMO.Proxy_Security_Pub_Join ().Response reçue et vérification cryptographique ok	Entrée appropriée peuplée dans le DSMO et le KeyDescriptor Appel PSMO.Security_Confirm(). Request (peut être retardée), ou Appel PSMO.Network_Information_ confirmation().Request	Joined (rattaché)
T3	Joined (rattaché)	SoftExpirationTime de la clé principale expiré	Appel PSMO.Security_New_Session() .Request avec le gestionnaire de sécurité	Updating (se mettant à jour)
T4	Updating (se mettant à jour)	DSMO.New_Key().Request(master_key) du gestionnaire de sécurité via le PSMO et vérification cryptographique ok	Enregistrer le matériel de clé principale et la politique. Définir le Key_ID de la session sur la valeur assignée par le gestionnaire de sécurité. Retourner une réponse DSMO.New_Key()	Clés superposées
T5	Joined (rattaché)	DSMO.New_Key().Request(ma ster_key) du gestionnaire de sécurité via le PSMO et vérification cryptographique ok	Enregistrer le matériel de clé principale et la politique. Définir le Key_ID de la session sur la valeur assignée par le gestionnaire de sécurité. Retourner une réponse DSMO.New_Key()	Clés superposées
T6	Clés superposées	ValidNotAfter de l'ancienne clé principale expiré.	Supprimer la clé principale expirée	Joined (rattaché)
T7	Updating (se mettant à jour)	Timeout or PSMO.Security_New_Session() .Response&& crypto check ok && SESSION_DENIED	Définir le moment du prochain essai	Joined (rattaché)
T8	Updating (se mettant à jour)	ValidNotAfter de la clé principale expirée	Supprimer la clé principale expirée	Provisioned (configuré)
T9	Joined (rattaché)	ValidNotAfter de la clé principale expirée	Supprimer la clé principale expirée	Provisioned (configuré)
T10	Joining (se rattachant)	Timeout	Diagramme d'états de réinitialisation	Provisioned (configuré)



Légende

Anglais	Français
Provisioned	Configuré
Joining	Se rattachant
Joined	Rattaché
Updating	Se mettant à jour
Overlapped keys	Clés superposées

Figure 49 – Transitions d'états d'appareil pour processus de rattachement et durée de vie d'appareil

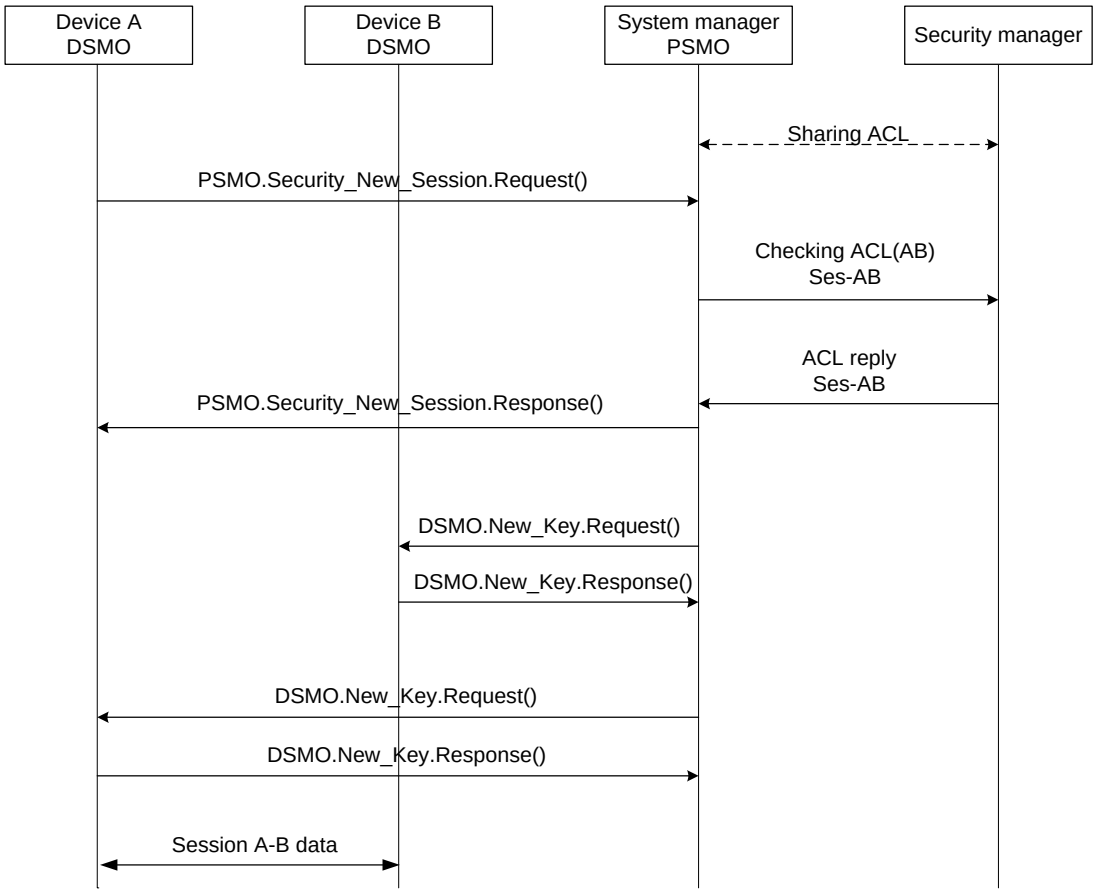
7.5 Etablissement de session

7.5.1 Généralités

L'établissement de session se produit à l'appui d'une communication sécurisée de bout en bout entre deux UAP. Le point d'extrémité d'une session est défini comme étant la concaténation de l'adresse IPv6Adress et du port de transport. Le gestionnaire de sécurité est responsable d'octroyer ou de refuser le support cryptographique utilisé pour établir la voie sécurisée de bout en bout entre les deux appareils.

7.5.2 Description

La Figure 50 fournit un exemple de haut niveau de l'établissement de session.



Légende

Anglais	Français
Device A DMSO	DMSO appareil A
Device B DMSO	DMSO appareil B
System manager PSMO	PSMO Gestionnaire de système
Security manager	Gestionnaire de sécurité
Sharing ACL	Partage ACL
Checking ACL(AB)	Vérification ACL(AB)
ACL reply	Réponse ACL
Session A-B data	Données de session A-B

Figure 50 – Exemple de haut niveau d'établissement de session

Dans l'exemple de haut niveau montré à la Figure 50, un UAP sur un appareil A établit une session avec un UAP sur un appareil B. Le DSMO de l'appareil A envoie la demande au gestionnaire de sécurité par l'intermédiaire de l'objet PSMO du gestionnaire de système. Le gestionnaire de système transmet alors la demande au gestionnaire de sécurité, qui authentifie la demande et peut accomplir une vérification pour vérifier si la session est autorisée. Si la session est accordée, le gestionnaire de sécurité génère une seule clé T pour les deux points d'extrémité, chiffre une copie pour l'appareil A et une autre copie pour l'appareil B, et transmet les messages au PSMO du gestionnaire de système. Le PSMO du gestionnaire de système peut alors envoyer la réponse à la demande de session. Le PSMO du gestionnaire de système appelle ensuite la méthode du DSMO pour ajouter une nouvelle clé T sur l'appareil A et l'appareil B.

L'établissement de session peut être initié par un appareil de terrain ou par le gestionnaire de sécurité/système.

- L'appareil de terrain initie un établissement de session en utilisant la méthode PSMO.Security_New_Session() dans le Tableau 80.
- Le gestionnaire de sécurité/système initie un établissement de session en utilisant la méthode de DSMO.New_Key() dans le Tableau 83.

Le gestionnaire de sécurité doit assigner les mêmes clé et CryptoKeyIdentifier parmi les appareils qui participent à la session sécurisée. Dans le cas des clés superposées, l'en-tête de sécurité doit acheminer le CryptoKeyIdentifier de la clé sélectionnée pour protéger la PDU. Au destinataire, l'appareil recherche le KeyDescriptor avec le CryptoKeyIdentifier spécifié dans l'en-tête de sécurité de la PDU entrante. Si la valeur de CryptoKeyIdentifier n'a pas de correspondance, l'appareil de réception ne sera pas capable de déchiffrer et/ou d'authentifier la PDU entrante.

7.5.3 Protection d'unité de données de protocole d'application en utilisant la clé principale

7.5.3.1 Généralités

La demande est faite à partir du DSMO de l'appareil au PSMO du gestionnaire de système agissant comme proxy du gestionnaire de sécurité. L'APDU doit être protégée avec l'utilisation du mécanisme de sécurité de PDU presque le même que la TL. La clé cryptographique doit être la clé principale, et le nonce doit être construit de la même manière que la TL en 7.3.3.7, mais la valeur de TAITimeRounded doit être utilisée pour le champ de création de temps TAI nominal. Voir le Tableau 363 pour les règles de codage appliquées aux valeurs de TAITimeRounded.

NOTE 1 Le processus de rattachement étant accompli à l'AL, il n'y a aucune sécurité à la TL au cours du processus en question, excepté des messages de confirmation.

La granularité du champ Time_Stamp étant exprimée en secondes, deux opérations cryptographiques utilisant la clé principale ne doivent pas être autorisées dans la même seconde.

NOTE 2 Si le débit de message utilisant la clé principale est supérieur au débit d'une fois par seconde, il y aura une collision de nonce.

7.5.3.2 Protection contre le rejet pour unité de données de protocole d'application protégée avec la clé principale

A la réception de l'APDU protégée avec la clé principale, la procédure de sécurité doit effectuer une vérification pour déceler tous les éventuels doublons de nonces avec un MIC valide dans le cache de nonces avec le KeyDescriptor correspondant. Si un nonce en double est détecté, la procédure doit rejeter la PDU avant de traiter l'ASDU, autrement la procédure doit stocker le nonce dans le cache de nonces dans le KeyDescriptor.

7.5.4 Méthodes d'objet proxy de gestion de sécurité relatives à l'établissement de session

Le Tableau 80 décrit la méthode Security_New_Session.

Tableau 80 – Méthode Security_New_Session

<table>
<tr><th colspan="5">Nom du type d'objet normalisé: Proxy security management object (PSMO, objet proxy de gestion de sécurité)</th></tr>
<tr><th colspan="5">Identificateur du type d'objet normalisé: 105</th></tr>
<tr><th>Nom de la méthode</th><th>ID de la méthode</th><th colspan="3">Description de la méthode</th></tr>
<tr><td rowspan="6">Security_New_Session</td><td>6</td><td colspan="3">Méthode d'utilisation du PSMO dans le gestionnaire de système pour envoyer une demande de session de sécurité et recevoir une réponse de session de sécurité</td></tr>
<tr><td colspan="4">Arguments d'entrée</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td>1</td><td>New_Session_Request</td><td>Security_New_Session_Request; voir Tableau 81</td><td>Demande de la nouvelle session de sécurité d'un appareil au gestionnaire de sécurité</td></tr>
<tr><td colspan="4">Arguments de sortie</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td>1</td><td>New_Session_Response</td><td>Security_New_Session_Response; voir Tableau 82</td><td>Réponse de la nouvelle session de sécurité du gestionnaire de sécurité à l'appareil demandeur, protégé par l'utilisation de la clé principale</td></tr>
</table>

La structure de données Security_New_Session_Request qui est utilisée pour former une demande de session est définie dans le Tableau 81.

Tableau 81 – Structure de données Security_New_Session_Request

Nom du type de données normalisé: Security_New_Session_Request		
Code du type de données normalisé: 420		
Nom de l'élément	**Identificateur de l'élément**	**Type de l'élément**
Originator_IPv6Adress	1	Type: IPv6Adress Classification: Static Accessibilité: Read/write
Originator_Port	2	Type: Unsigned16 Classification: Static Accessibilité: Read/write
Destination_IPv6Adress	3	Type: IPv6Adress Classification: Static Accessibilité: Read/write
Destination_Port	4	Type: Unsigned16 Classification: Static Accessibilité: Read/write
Algorithm_Identifier	5	Type: Unsigned8 Classification: Static Accessibilité: Read Only Valeur par défaut: 1 = AES_CCM*
Protocol_Version	6	Type: Unsigned8 Classification: Static Accessibilité: Read Only Valeur par défaut: 1 = IEC 62734 Ed. 1.0 (la présente norme, par exemple)
Security_Control	7	Type: Unsigned8 Classification: Static Accessibilité: Read/write
Crypto_Key_Identifier	8	Type: Unsigned8 Classification: Static Accessibilité: Read/write
Time_Stamp	9	Type: TAITimeRounded Classification: Static Accessibilité: Read Only
MIC	10	Type: OctetString (SIZE = 4, 8, 16) Classification: Static Accessibilité: Read Only

Cette structure de données est constituée d'une section en texte clair uniquement protégée en utilisant la clé principale partagée entre le demandeur de la session et le gestionnaire de sécurité. L'adresse EUI64Address du demandeur doit être utilisée dans la construction de nonce pour protéger cette structure.

- Originator_IPv6Adress doit être l'adresse IPv6Adress du premier appareil d'extrémité (habituellement la source) dans la session.
- Originator_Port doit être le port T du premier UAP d'extrémité (habituellement la source) dans la session.

- Destination_IPv6Adress doit être l'adresse IPv6Adress du second appareil d'extrémité (habituellement la destination) dans la session.
- Destination_Port doit être le port T du second UAP d'extrémité (habituellement la destination) dans la session.
- Algorithm_Identifier définit l'algorithme et le mode de fonctionnement pris en charge dans cette session. Dans la version courante, il est mis à 0x1 = AES_CCM*.
- La version de protocole identifie le protocole utilisé pour cette association de sécurité. Dans la présente norme, cet octet doit être 0x01.
- Security_Control doit être tel que défini en 7.3.1.2. Le niveau de sécurité est sélectionné dans MIC-32, MIC-64 et MIC-128 avec le niveau de sécurité de clé principale assigné dans le processus de rattachement. Le CryptoKeyIdentifierMode doit être '01' correspondant à la taille du champ Crypto_Key_Identifier de 1 octet.
- Crypto_Key_Identifier doit être le Crypto_Key_Identifier de la clé principale utilisée dans la protection de cette structure.
- Time_Stamp doit être la représentation tronquée de 32 bits du temps TAI utilisée dans la construction de nonce T.
- MIC doit être le code d'intégrité généré par le calcul de l'AES_CCM*. La taille du MIC est assignée dans le champ Security_Control .

Le nonce utilisé pour générer le MIC est formé conformément au Tableau 57 avec:

– EUI64Address: Adresse EUI64Address de l'appareil transmettant le message de demande Security_New_Session.

– Temps TAI nominal: Le champ Time_Stamp dans le message Security_New_Session Request.

La structure de données Security_New_Session_Response qui est utilisée pour former une nouvelle réponse de session est définie dans le Tableau 82.

Tableau 82 – Structure de données Security_New_Session_Response

Nom du type de données normalisé: Security_New_Session_Response		
Code du type de données normalisé: 421		
Nom de l'élément	**Identificateur de l'élément**	**Type de l'élément**
Status	1	Type: Unsigned8 Classification: Static Accessibilité: Read/write
Security_Control	2	Type: Unsigned8 Classification: Static Accessibilité: Read/write
Crypto_Key_Identifier	3	Type: Unsigned8 Classification: Static Accessibilité: Read/write
Time_Stamp	4	Type: TAITimeRounded Classification: Static Accessibilité: Read Only
MIC	5	Type: OctetString (SIZE = 4, 8, 16) Classification: Static Accessibilité: Read Only

Les champs comprennent:

- Status doit être le statut de la session où 0x1 = SECURITY_SESSION_GRANTED et 0x0 = SECURITY_SESSION_DENIED.
- Security_Control doit être tel que défini en 7.3.1.2. Le niveau de sécurité est sélectionné dans MIC-32, MIC-64 et MIC-128 avec le niveau de sécurité de clé principale assigné au cours du processus de rattachement. Le CryptoKeyIdentifierMode doit être '01' correspondant à une taille du champ CryptoKeyIdentifier de 1 octet.
- Crypto_Key_Identifier doit être le Crypto_Key_Identifier de la clé principale utilisée dans la protection de cette structure.
- Time_Stamp doit être la représentation tronquée de 32 bits du temps TAI utilisée dans la construction de nonce.
- MIC doit être le code d'intégrité généré par le calcul de l'AES_CCM*. La taille du MIC est assignée dans le champ Security_Control.

Le nonce utilisé pour générer le MIC est formé conformément au Tableau 57 avec:

– EUI64Address: Adresse EUI64Address de l'appareil transmettant le message de demande Security_New_Session.
– Temps TAI nominal: Le champ Time_Stamp issu du message Security_New_Session Request.

Si la session est accordée, le gestionnaire de sécurité par le truchement du PSMO du gestionnaire de système doit appeler la méthode New_Key du DSMO définie en 7.6.3 pour écrire une nouvelle clé T dans les appareils spécifiés dans la demande de session.

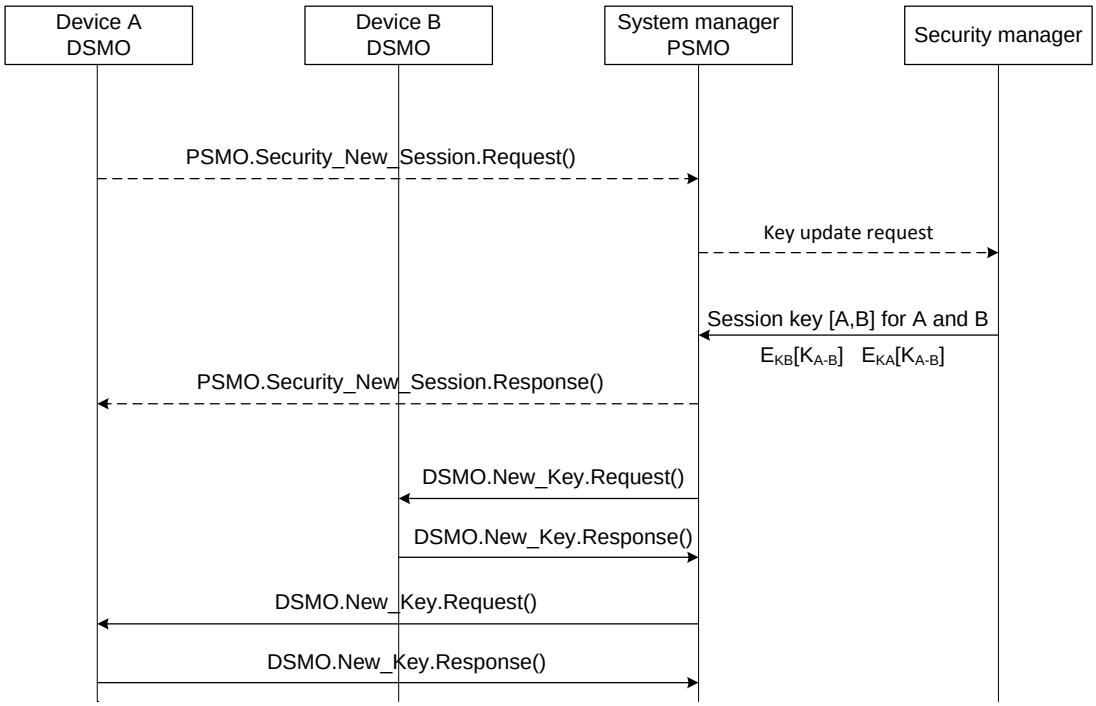
7.6 Mise à jour de clé

7.6.1 Généralités

Les clés T ont une durée de vie limitée et sont mises à jour périodiquement pour assurer que la session est maintenue vivante. Le processus de mise à jour de clé peut être initié par un appareil, même s'il convient qu'il soit poussé à partir du gestionnaire de sécurité entre le SoftExpirationTime et le HardExpirationTime d'une clé T.

7.6.2 Description

Le processus de mise à jour de clé est résumé à la Figure 51. Une TLE peut demander que le gestionnaire de sécurité mette à jour une clé T. Le gestionnaire de sécurité émet alors un appel au DSMO des TLE de point d'extrémité, par l'intermédiaire du PSMO du gestionnaire de système, pour mettre à jour la clé T pour ces TLE. Chaque message de ce type est protégé avec la clé principale active partagée entre le gestionnaire de sécurité et le DSMO de la TLE spécifique.



Légende

Anglais	Français
Device A DMSO	DMSO appareil A
Device B DMSO	DMSO appareil B
System manager PSMO	PSMO gestionnaire de système
Security manager	Gestionnaire de sécurité
Key update request	Demande de mise à jour de clé
Session key [A, B] for A and B	Clé de session [A, B] pour A et B

Figure 51 – Vue d'ensemble du protocole de mise à jour de clé

Une TLE participant à une session peut initier le processus de mise à jour de clé en faisant un appel à la méthode de Security_New_Session du PSMO. La demande est transmise du PSMO du gestionnaire de système vers le gestionnaire de sécurité, qui authentifie la demande en utilisant la clé principale de l'appareil demandeur. Si la vérification est couronnée de succès, le gestionnaire de sécurité reconnaît que la session existe déjà et continue simplement avec le protocole de mise à jour de clé exactement comme si le SoftExpirationTime de la clé T a expiré. La construction de nonce pour protéger l'APDU utilisant la clé principale est décrite en 7.5.3.

Si le SoftExpirationTime d'une clé T active a passé, le gestionnaire de sécurité doit appeler la méthode New_Key sur le DSMO des appareils d'extrémité pour écrire une nouvelle clé et les politiques d'accompagnement.

Les méthodes de Key Update (mise à jour de clé) peuvent également être utilisées pour mettre à jour la clé de DL et la clé principale.

7.6.3 Méthodes de l'objet de gestion de sécurité d'appareil relatives à la mise à jour de clé T

Le Tableau 83 décrit la méthode New_Key.

Tableau 83 – Méthode New_Key

Nom du type d'objet normalisé: Device security management object (DSMO, objet de gestion de sécurité d'appareil)				
Identificateur du type d'objet normalisé: 125				
Nom de la méthode	**ID de la méthode**	**Description de la méthode**		
New_Key	1	Méthode d'utilisation du DSMO dans l'appareil pour l'envoi d'une clé de sécurité protégée et des politiques d'accompagnement		
	Arguments d'entrée			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	Key_And_Policies	Security_Key_and_Policies; voir Tableau 84	Les clés de sécurité et les politiques doivent être authentifiées, décryptées et stockées par un appareil participant à la session
	Arguments de sortie			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	Key_Update_Status	Security_Key_Update_Status; voir Tableau 85	Statut de mise à jour de la clé, authentifié avec la clé principale

La structure de données Security_Key_and_Policies qui est utilisée pour former une demande de nouvelle clé T (New T-key) est définie dans le Tableau 84.

Tableau 84 – Structure de données Security_Key_and_Policies

Nom du type de données normalisé: Security_Key_and_Policies		
Code du type de données normalisé: 422		
Nom de l'élément	**Identificateur de l'élément**	**Type de l'élément**
Key_Policy	1	Type: OctetString (voir Tableau 88) Classification: Static Accessibilité: Read Only
End_Port_Source (élidé pour un DL ou une clé principale)	2	Type: Unsigned16 Classification: Static Accessibilité: Read/write
EUI64_remote (élidé pour un DL, une adresse EUI64Address du gestionnaire de sécurité pour la clé principale)	3	Type: EUI64Address Classification: Static Accessibilité: Read/write
128_Bit_Address_remote (élidé pour un DL ou une clé principale)	4	Type: IPv6Address Classification: Static Accessibilité: Read/write
End_Port_remote (élidé pour un DL ou une clé principale)	5	Type: Unsigned16 Classification: Static Accessibilité: Read/write
Algorithm_Identifier	6	Type: Unsigned8 Classification: Static Accessibilité: Read Only Valeur par défaut: 1 = AES_CCM*
Security_Control	7	Type: Unsigned8 Classification: Static Accessibilité: Read/write
Crypto_Key_Identifier	8	Type: Unsigned8 Classification: Static Accessibilité: Read/write
Time_Stamp	9	Type: TAITimeRounded Classification: Static Accessibilité: Read Only
New_Key_ID	10	Type: Unsigned8 Classification: Static Accessibilité: Read Only
Key_Material	11	Type: SymmetricKey Classification: Static Accessibilité: Read Only
MIC	12	Type: OctetString (SIZE = 4, 8, 16) Classification: Static Accessibilité: Read Only

Cette structure de données est constituée d'une section en texte clair uniquement protégée en utilisant la clé principale partagée entre le demandeur de la session et le gestionnaire de sécurité. L'adresse EUI64Address du demandeur doit être utilisée dans la construction de nonce pour protéger cette structure.

Les champs comprennent:

- 128_Bit_Address_remote doit être l'adresse IPv6Adress de la TLE de point d'extrémité distante dans cette session. Dans le cas de la clé D ou de la clé principale, ce champ doit être élidé.
- End_Port_remote doit être le port T de l'UAP de point d'extrémité distante dans cette session. Dans le cas de la clé D ou de la clé principale, ce champ doit être élidé.
- Algorithm_Identifier définit l'algorithme et le mode de fonctionnement pris en charge dans cette session. Dans la version courante, il est mis à 0x1 = AES_CCM*.
- Security_Control doit être tel que défini en 7.3.1.2. Le niveau de sécurité est sélectionné dans ENC-MIC-32, ENC-MIC-64 et ENC-MIC-128 avec le niveau de sécurité de clé principale assigné dans le processus de rattachement. Le CryptoKeyIdentifierMode doit être '01' correspondant à la taille du champ Key Index de 1 octet.
- Crypto_Key_Identifier doit être le Crypto_Key_Identifier de la clé principale utilisée dans la protection de cette structure.
- Time_Stamp doit être la représentation tronquée de 32 bits du temps TAI utilisée dans la construction de nonce.
- Key_Policy doit être tel que décrit dans le Tableau 88 et peuplé par le gestionnaire de sécurité sur la base de ses politiques de sécurité pour cette session.
- New_Key_ID doit être le Crypto_Key_Identifier de 8 bits assigné pour ce support de clé par le gestionnaire de sécurité.
- Key_Material doit être une clé symétrique pour cette session.
- MIC doit être le code d'intégrité généré par le calcul de l'AES_CCM*. La taille du MIC est spécifiée dans le champ Security_Control.

Le niveau de sécurité pour la clé principale doit être mis au moins à la force de la clé la plus élevée utilisée.

La structure de données Security_Key_and_Policies est protégée par AES-CCM* avec les paramètres suivants:

- partie authentification: élément 1..10
- partie cryptage: élément 11
- clé: clé principale
- nonce: formé avec la structure du Tableau 57 avec:
  - EUI64Address: Adresse EUI64Address du gestionnaire de sécurité
  - temps TAI nominal: Elément Time Stamp acheminé dans Security_Key_and_Policies

A la réception de l'appel de la méthode DSMO.New_Key().Request, le DSMO de l'appareil d'extrémité doit déchiffrer et faire une vérification d'intégrité sur la PDU en utilisant la même étape de traitement de PDU entrante telle que définie dans la TL (voir 7.3.3.9) avec le nonce construit avec l'adresse EUI64Address du gestionnaire de sécurité et les 32 bits de temps inclus dans la PDU. La clé utilisée doit être la clé principale courante telle qu'identifiée par le CryptoKeyIdentifier.

A l'achèvement réussi de la vérification, le KeyDescriptor approprié doit être peuplé en utilisant les champs dans la structure de données Security_Key_and_Policies. Dans la présente version, l'émetteur est toujours le gestionnaire de sécurité.

Le DSMO doit alors produire un message de statut tel que défini dans le Tableau 85 pour notifier au gestionnaire de sécurité le statut de l'appel de méthode.

La structure de données Security_Key_Update_Status qui est utilisée pour former la réponse à la demande de mise à jour de clé est définie dans le Tableau 85.

Tableau 85 – Structure de données Security_Key_Update_Status

Nom du type de données normalisé: Security_Key_Update_Status		
Code du type de données normalisé: 423		
Nom de l'élément	**Identificateur de l'élément**	**Type de l'élément**
Status	1	Type: Unsigned8 Classification: Static Accessibilité: Read/write
Security_Control	2	Type: Unsigned8 Classification: Static Accessibilité: Read/write
Crypto_Key_Identifier	3	Type: Unsigned8 Classification: Static Accessibilité: Read/write
Time_Stamp	4	Type: TAITimeRounded Classification: Static Accessibilité: Read Only
MIC	5	Type: OctetString (SIZE = 4, 8, 16) Classification: Static Accessibilité: Read Only

Les champs comprennent:

- Status doit être le statut de la session où 0x1 = SECURITY_KEY_UPDATE_FAILURE et 0x0 = SECURITY_KEY_UPDATE_SUCCESS.
- Security_Control doit être tel que défini en 7.3.1.2. Le niveau de sécurité est sélectionné dans MIC-32, MIC-64 et MIC-128 avec le niveau de sécurité de clé principale assigné au cours du processus de rattachement. Le CryptoKeyIdentifierMode doit être '01' correspondant à une taille du champ CryptoKeyIdentifier de 1 octet.
- Crypto_Key_Identifier doit être le Crypto_Key_Identifier de la clé principale utilisée dans la protection de cette structure.
- Time_Stamp doit être la représentation tronquée de 32 bits du temps TAI utilisée dans la construction de nonce.
- MIC doit être le code d'intégrité généré par le calcul de l'AES_CCM*. La taille du MIC est assignée dans le champ Security_Control.

Le nonce utilisé pour générer le MIC est formé conformément au Tableau 57 avec:

– EUI64Address: Adresse EUI64Address de la DLE émettant le message de réponse New_Key.

– Temps TAI nominal: L'élément Time_Stamp issu du message Security_Key_Update_Status.

7.6.4 Rétablissement après défaillance

7.6.4.1 Généralités

A un point quelconque au cours de l'établissement de session ou du processus de mise à jour de clé, il y a une possibilité qu'une PDU soit abandonnée. Dans ce cas, il convient que le système soit capable de récupérer et continuer. Les définitions d'états et les transitions d'états suivantes décrivent les grandes lignes du mécanisme de rétablissement, avec les effets secondaires déclenchés.

7.6.4.2 Etats de la clé T et de la clé D

Les états de clé T et de clé D comprennent:

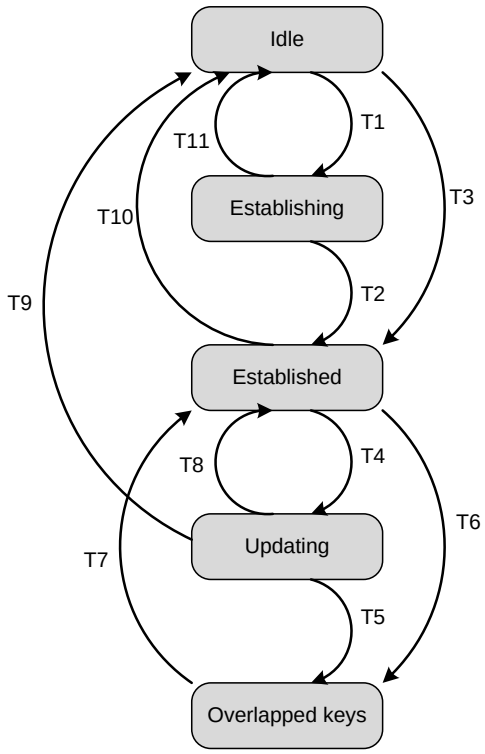
- Idle (inactif): Aucune clé, n'est pas en train d'obtenir la clé T courante.
- Establishing (s'établissant): Aucune clé, est en train d'obtenir la clé T courante.
- Established (établi): Ayant la clé courante, n'est pas en train d'obtenir la clé suivante.
- Updating (se mettant à jour): Ayant la clé courante, est en train d'obtenir la clé suivante.
- Overlapped (superposé): Ayant la clé courante et la clé suivante

7.6.4.3 Transition d'état de clé T et de clé D

Les transitions d'état montrées dans le Tableau 86 et à la Figure 52 montrent l'état de l'appareil initiant une récupération de clé T ou de clé D, et celui de son homologue acceptant la demande. Toutes les deux commencent dans l'état "idle", et toutes les deux finissent dans l'état "established" avec une session ou clé D valide et les éléments cryptographiques pertinents.

Tableau 86 – Transition d'états de clé T et de clé D

Transition	Etat actuel	Evénement(s)	Action(s)	Etat suivant
T1	Inactif	DSMO a demandé une nouvelle session	Appel de la méthode suivante sur le gestionnaire de système: PSMO.Security_New_Session. Request()	Establishing (s'établissant)
T2	Establishing (s'établissant)	PSMO.Security_New_Session() .Response && crypto check ok	Enregistrer le support de clé, l'indice de politique et d'emplacement, l'adresse distante, le port distant, l'adresse locale, le port local, selon les besoins	Established (Etablie)
T3	Inactif	DSMO.New_Key().Request du gestionnaire de sécurité via le PSMO et vérification cryptographique ok	Enregistrer le support de clé, l'index de politique et d'emplacement, l'adresse distante, le port distant, l'adresse locale, le port local, selon les besoins. Retourner une DSMO.New_Key.Response()	Established (Etablie)
T4	Established (Etablie)	Session ou SoftExpirationTime de la clé D expirés	Appel de la méthode suivante sur le gestionnaire de système: PSMO.Security_New_Session. Request()	Updating (se mettant à jour)
T5	Updating (se mettant à jour)	DSMO.New_Key().Request du gestionnaire de sécurité via le PSMO et vérification cryptographique ok	Enregistrer le support de clé, l'index de politique et d'emplacement, l'adresse distante, le port distant, l'adresse locale, le port local, selon les besoins. Retourner une DSMO.New_Key.Response()	Clés superposées
T6	Established (Etablie)	DSMO.New_Key().Request du gestionnaire de sécurité via le PSMO et vérification cryptographique ok	Stocker les nouvelles clés dans la mémoire	Clés superposées
T7	Clés superposées	ValidNotAfter de la session courante ou de la clé D expirés	Supprimer la clé expirée	Established (Etablie)
T8	Updating (se mettant à jour)	Timeout OR PSMO.Security_New_Session. Response() && crypto check ok && SESSION_DENIED	Définir le moment du prochain essai	Established (Etablie)
T9	Updating (se mettant à jour)	ValidNotAfter de la dernière session ou de la clé D expiré	Supprimer la clé expirée	Inactif
T10	Established (Etablie)	ValidNotAfter de la dernière session ou de la clé D expirés	Supprimer la clé expirée	Inactif
T11	Establishing (s'établissant)	Timeout	Diagramme d'états de réinitialisation et définir le moment du prochain essai si nécessaire	Inactif



Légende

Anglais	Français
Idle	Inactif
Establishing	S'établissant
Established	Etabli
Updating	Se mettant à jour
Overlapped keys	Clés superposées

NOTE 1 Si un appareil reçoit la demande DSMO.New_Key() alors qu'il est dans l'état Updating ou Overlapped, l'appareil rejette la clé principale, qui n'est pas utilisée pour chiffrer la nouvelle clé principale.

NOTE 2 Si l'appareil reçoit par une demande de DSMO.New_Key() une nouvelle clé principale qui a été chiffrée en utilisant une clé principale inconnue, l'appareil est capable d'interroger le gestionnaire de sécurité pour la clé principale nécessaire pour le déchiffrement avec la demande de PSMO.New_Session_Request(), afin de resynchroniser les clés principales. Le gestionnaire de sécurité a les informations nécessaires pour inférer, sélectionner et utiliser la clé principale appropriée.

Figure 52 – Etablissement de clé d'appareil et transition d'états de mise à jour de clé

7.7 Fonctionnalité du rôle de gestionnaire de sécurité

7.7.1 Proxy security management object (objet proxy de gestion de sécurité)

Les attributs du PSMO sont donnés dans le Tableau 87.

Tableau 87 – Attributs du PSMO dans le gestionnaire de système

Nom du type d'objet normalisé: Proxy security management object (PSMO, objet proxy de gestion de sécurité)				
Identificateur du type d'objet normalisé: 105				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Réservé pour les éditions futures de la présente norme	1..63	-	-	-

Note Un objet a des attributs, ou des méthodes, ou les deux. Cet objet n'utilise que des méthodes normales; aucun attribut normal n'a été identifié comme étant nécessaire. Par conséquent, dans cette édition de la présente norme, le Tableau 87 est vide (null).

7.7.2 Autorisation des appareils de réseau et génération ou dérivation de clés principales initiales

Le gestionnaire de sécurité maintient une base de données contenant:

- une liste d'appareils dont les authentifiants ont été établis par un processus de configuration ou à la première tentative de rattachement au réseau, et qui n'ont pas été révoqués; et
- une liste de clés de rattachement valides et de leurs durées de vie associées qui ont été émises aux appareils d'agent de configuration, qui pourrait être fournie par de nouveaux appareils qui tentent de rejoindre le réseau.

Lorsqu'un nouvel appareil tente de rejoindre le réseau et sa demande vient du gestionnaire de sécurité par l'intermédiaire du PSMO du gestionnaire de système selon la procédure décrite en 7.4.5, le gestionnaire de sécurité examine les deux premières listes pour une décision rapide d'acceptation/rejet. Autrement, la procédure doit être telle que décrite en 7.4.6.

7.7.3 Interaction avec des objets de gestion de sécurité d'appareil

Un gestionnaire de sécurité interagit avec le DSMO d'un appareil par l'intermédiaire du PSMO:

- au cours du processus de rattachement;
- lorsqu'il distribue de nouvelles clés;
- lorsqu'il reçoit de nouvelles clés qui ont été établies par d'autres appareils par le biais de l'utilisation de protocoles d'agrément de clé;
- au cours d'un processus de rattachement comme décrit en 7.4.4; et
- au cours de la récupération de clé lorsqu'un nouveau gestionnaire de sécurité réseau en remplace un qui a échoué.

NOTE Seul le gestionnaire de sécurité génère de nouvelles clés une fois qu'un appareil est rattaché au réseau.

7.7.4 Gestion de clés opérationnelles

7.7.4.1 Généralités

Le gestionnaire de sécurité maintient la clé principale courante et les attributs associés de génération de clés et de politique pour chaque appareil qu'il gère.

Toutes les clés symétriques sont maintenues dans le stockage opérationnel du gestionnaire de sécurité et il convient que dans les mises en œuvre de plus haute sécurité celui-ci soit physiquement protégé au sein du module cryptographique du gestionnaire de sécurité.

Lorsque les appareils participants ne mettent pas tous les deux en œuvre le jeu de primitives cryptographiques asymétriques spécifiées en 7.4.6, qui est une option de construction pour chaque appareil, le gestionnaire de sécurité peut également générer des clés de données symétriques de secrets partagés pour les associations de DL et de TL en monodiffusion.

NOTE De nombreux appareils n'ont pas une source de bits aléatoires d'entropie élevée. Sans une telle source, n'importe quelle composante de clé générée par l'appareil est potentiellement sensible aux interférences.

7.7.4.2 Archivage de clé

La réglementation ou la politique peut exiger que les clés soient archivées pour permettre le déchiffrement, simultané ou ultérieur, de la messagerie chiffrée.

7.7.4.3 Récupération de clé

Il convient que le gestionnaire de sécurité prenne en charge deux formes de récupération de clé:

- récupération par un appareil de terrain qui a perdu des clés qui étaient maintenues dans le stockage volatil (RAM, par exemple) en raison d'une panne de courant ou d'une erreur mémoire non corrigée; et
- récupération par un nouveau gestionnaire de sécurité réseau des clés opérationnelles actuellement en utilisation dans le réseau.

Chaque appareil de terrain qui prend en charge la cryptographie à clés asymétriques doit conserver dans le stockage non volatil:

– son adresse EUI64Address; et
– sa paire de clés publique/privée, avec cette dernière comme un certificat signé si disponible.

Chaque appareil de terrain qui ne prend en charge que la cryptographie à clés symétriques doit conserver dans le stockage non volatile:

– son adresse EUI64Address;
– sa clé de rattachement; et
– sa clé de rattachement courante et ses informations de codage connexes, s'il a précédemment été un membre du réseau.

Toutes les autres informations de codage opérationnelles peuvent être conservées dans le stockage volatil, susceptibles d'être perdues en cas de panne de courant d'appareil ou corruption de mémoire, car ces informations peuvent être régénérées par un gestionnaire de sécurité une fois que le gestionnaire de sécurité a déterminé l'adresse EUI64Address de l'appareil.

7.7.4.4 Administration de politique de sécurité

Les principales options de déploiement affectant la politique de sécurité à l'échelle du réseau sont sélectionnées pendant l'installation initiale du gestionnaire de sécurité pour un réseau. D'autres options de déploiement de politique peuvent être sélectionnées à un moment ultérieur.

7.8 Politiques de sécurité

7.8.1 Définition de politique de sécurité

Dans la présente norme, la politique de sécurité est définie comme étant une combinaison des paramètres suivants:

- Key Type (type de clé) défini dans le Tableau 89;

- Key Usage (utilisation de clé) définie dans le Tableau 90;
- Key Lifetime (durée de vie de clé) définie en 7.2.2.4; et
- Security Level (niveau de sécurité) défini en 7.3.1.1.

Les clés sont distribuées avec les paramètres spécifiés ci-dessus de façon explicite ou reconstruits de façon implicite au destinataire. Un KeyDescriptor correspondant est généré avec ces paramètres.

7.8.2 Etendue de politique

Les politiques de sécurité limitent les choix de sécurité que les programmes et appareils individuels peuvent faire. Ces politiques existent aux niveaux suivants:

- à l'échelle de sous-réseau, à travers tous les appareils participant à un sous-réseau D donné, qui peut englober le système réseauté entier;
- à l'échelle d'appareil, à travers tous les programmes d'application et les couches communications d'appui au sein de l'appareil;
- à l'échelle de clé, à travers toutes les PDU sécurisées avec une clé donnée; et
- à l'échelle de liaison, à travers toutes les PDU émises sur une connexion donnée définie par une source et une destination, qui peuvent inclure des ports d'UAP, fournissant ainsi des politiques à l'échelle d'UAP, à travers toutes les invocations de service par une application donnée.

Certaines politiques à l'échelle du système doivent être établies avant que l'exploitation du système ne commence; d'autres peuvent être modifiées de façon dynamique pendant que le système fonctionne, sans interrompre les sessions en cours.

7.8.3 Choix de politiques de sécurité non contraintes

Des choix de politique de sécurité de système peuvent être faits pendant l'exploitation du système. La nouvelle politique entrera en vigueur avec le prochain recodage des appareils affectés effectué par un gestionnaire de sécurité. Ainsi, l'opération avec une clé symétrique donnée a toujours un jeu fixé d'attributs.

7.8.4 Structures de politique

Le format des politiques est présenté dans le Tableau 88.

Tableau 88 – Structure de champ "policy"

Octet	Bits							
	7	6	5	4	3	2	1	0
0	Key_Type			Key_Usage			Granularité	
1..4	ValidNotBefore nominal							
5..6 (7, 8 opt)	HardLifeSpan							
9	Security_Level			Réservé				

Les champs comprennent:

- Key_Type: type de clé défini dans le Tableau 89.
- Key_Usage: utilisation de clé définie dans le Tableau 90.
- Granularité: unité en laquelle la Nominal HardLifeSpan est interprétée, telle que définie dans le Tableau 91.

- ValidNotBefore nominal en secondes: temps TAI absolue sous la forme TAITimeRounded, où le destinataire de la clé peut commencer à utiliser la clé.
- HardLifeSpan: durée pendant laquelle cette clé est valide. La durée valide de la clé commence à ValidNotBefore. Si le champ ValidNotAfter est rempli avec 0x00 pour une granularité quelconque, cette clé a une durée de vie infinie et la clé n'expirera donc jamais. A moins que ValidNotAfter soit infini, la durée réelle de la KeyHardLifeSpan ne doit excéder 48,5 jours (voir 7.3.2.4.10) dans aucune granularité.
- Security_Level: niveau de sécurité pour chaque clé, tel que défini dans le Tableau 35.
- Reserved: il convient que le champ Reserved soit 0.

Les valeurs possibles pour les types de clé sont présentées dans le Tableau 89.

Tableau 89 – Key_Type

Valeur Key_type	Description
0	Réservé
1	Support de codage de clé symétrique, cryptage
2	Certificat manuel ECC
3	Certificat implicite ECC
4..7	Réservé

Les valeurs possibles pour l'utilisation de la clé sont présentées dans le Tableau 90.

Tableau 90 – Key_Usage

Valeur Key_Usage	Description
0	Clé de groupe pour traitement de PDU (clé D par exemple)
1	Clé de lien pour traitement de PDU (clé T par exemple)
2	Clé principale pour l'établissement de session
3	Clé de rattachement
4	Clé publique pour plan ECMQV
5	Clé racine CA pour plan ECQV
6	Réservé
7	Clé non secrète globale fixe

La granularité de la HardLifeSpan dans la politique de clé est présentée dans le Tableau 91.

Tableau 91 – Granularité

Granularité	Unité de temps SI [a]	Nom commun	Facteur d'échelle	HardLifeSpan (octets)
0	s	Seconde	1 s	4
1	Min	Minute	60 s	3
2	h	Heure	3 600 s	2
3	d	Day	86 400 s	2

[a] Bien que "s" soit la seule unité de temps SI officielle, l'utilisation des autres unités listées dans la seconde colonne est acceptée avec le système SI.

Les politiques suivantes doivent être disponibles à un réseau spécifié par la présente norme. La variable *k* indique une variable qui peut être établie par le gestionnaire de sécurité d'un réseau donné.

- Alertes et journalisation:

  Un appareil garde la trace du nombre de calculs cryptographiques défaillants sur une période de temps. Si un seuil configurable est franchi, une alerte est générée. Les alertes incluent "taux d'échecs de données DPDU excédé", "taux d'échecs de TPDU excédé", et "taux d'échecs de mise à jour de clé dépassé". Voir les alertes en 7.11.4.

- Politique d'appareil:

  L'authentification D doit toujours être active avec une taille d'étiquette d'authentification de 32 bits. La clé par défaut utilisée par un appareil se joignant est la K_global bien connue, utilisée pour détecter seulement des erreurs aléatoires. Une clé secrète protège l'APDU de niveau plus haut au cours d'un rattachement sécurisé. Voir 7.4.

  NOTE 1 La taille du MIC de DL est spécifiée en 7.3.1.1 et les contraintes pour le DMIC sont spécifiées en 7.3.2.

- Politique de clé:

  La liaison, l'association de sécurité et la politique de PDU sont appliquées par le biais de la politique de clé. Tous les utilisateurs d'une clé donnée doivent avoir la même politique; voir Tableau 88. Les éléments configurables incluent:

  – les types de clés; voir Tableau 89;
  – la granularité du temps TAI utilisé dans la durée de vie de la clé; voir Tableau 91;
  – la taille de MIC (32 bits, 64 bits ou 128 bits); voir Tableau 35;
  – la taille de DMIC de 32 bits, 64 bits ou 128 bits, mise à 32 par défaut, définie dans le DMXHR; voir 7.3.2.2;
  – la taille de TMIC de 0, 32, 64, ou 128 est mise à 0 si la sécurité est désactivée, et mise à 32 si la sécurité est activée. Durée de vie Soft; voir Tableau 93;
  – chiffrement de charge utile activé/désactivé;
  – chiffrement de DL activé/désactivé, désactivé par défaut, mis dans le DMXHR; voir 7.3.2.2;
  – chiffrement de TL activé/désactivé, désactivé si la sécurité est désactivée, activé si la sécurité est activée;
  – HardLifeSpan (voir Tableau 88), exprimée comme une valeur absolue de temps TAI, limitée par la durée maximale à partir du temps de génération de clé qui empêchera un passage par zéro du nonce;

    NOTE 2 Cette question est liée à la granularité du temps de TAI dans le nonce (voir 6.3.10); 48,5 jours si utilisé à 1 024 PDU/s.

  – key originator: adresse EUI64Address du générateur d'une clé donnée (habituellement le gestionnaire de sécurité) qui doit établir la politique pour une clé donnée;
  – sessions autorisées dans le gestionnaire de sécurité.

- Politique de contrôle d'accès:

  La fonction de contrôle d'accès pour un établissement de session n'est exigée que dans le gestionnaire de sécurité. Le gestionnaire de sécurité décide d'accorder ou de refuser une session dans la phase d'établissement de session. Le résultat est retourné par la méthode PSMO.Security_New_Session(). Si un gestionnaire de sécurité a à la fois une liste Allowed et une liste Disallowed, le gestionnaire de sécurité peut indiquer laquelle de ces deux listes est prioritaire.

  – Liste Allowed: Le gestionnaire de sécurité peut avoir une liste d'appareils autorisés identifiés par des informations valides (par exemple: adresse EUI64Address et nom de TAG) énumérées dans le Tableau 372.

– Liste Disallowed: Le gestionnaire de sécurité peut avoir une liste d'appareils autorisés identifiés par des informations valides (par exemple: adresse EUI64Address et nom de TAG).

7.9 Fonctions de sécurité disponibles à l'AL

7.9.1 Paramètres sur les demandes de service de transport qui se rapportent à la sécurité

Les UAP sont autorisés à établir des associations d'applications de façon dynamique en demandant une session devant être établie. Voir 6.3.11.2.5.2. Après qu'une session est établie, toutes les communications issues de cet UAP avec ces homologues sont traitées en toute sécurité jusqu'à une période de non-utilisation ou jusqu'à ce qu'un besoin de réutiliser le stockage pour les informations d'état de sécurité amène la TL à mettre fin à l'association de sécurité de transport antérieure. Si une demande ultérieure de service de transport émise de l'UAP vers ces homologues se produit après que l'association de sécurité de transport a été interrompue, cette demande ultérieure doit être traitée comme une nouvelle demande, conduisant à une nouvelle association de sécurité de transport.

Il n'y a intentionnellement aucune aptitude de porter des TPDU non sécurisées sur l'association de sécurité de transport une fois qu'elle a été établie, car un tel mécanisme serait trivialement facile à attaquer en altérant simplement des TPDU authentifiées sélectionnées pour indiquer qu'elles n'utilisaient aucune authentification.

Pour prendre en charge des services d'AL sans état, la politique la moins récemment utilisée peut être appliquée par des couches sous-jacentes pour recycler tous les éventuels engagements de ressource (état de connexion de sécurité, par exemple) qu'elles pourraient prendre.

Cela aiderait à l'exploitation efficace du système de sécurité si chaque demande de service de transport sur une association avait la capacité de faire allusion à l'intervalle prévu avant la prochaine utilisation de l'association. Une telle allusion fournit des conseils quant à la gestion des connexions implicites de sécurité de transport exigées pour des communications de transport sécurisées, permettant un placement intelligent en cache des connexions de sécurité établies et réduisant au maximum la dégradation qui se produit lorsqu'une connexion de sécurité implicite est fermée puis rouverte après qu'une nouvelle clé a été établie à tous les participants d'association.

Les niveaux de sécurité permis (voir 7.3.1.1) sur une demande de service de transport sont:

- chiffrement de charge utile de couche supérieure de TPDU: on/off;
- authentification de la TPDU avec un TMIC de taille 32 bits, 64 bits ou 128 bits.

Dans un API, ceux-ci peuvent être acheminés conjointement sous la forme d'un seul nombre entier signé (un Integer8, par exemple), où le signe a été utilisé pour désigner le chiffrement (–) ou pas (+), et la grandeur a été utilisée pour spécifier la taille demandée nn, avec la valeur zéro représentant une demande pour "aucune authentification" et "aucun chiffrement".

7.9.2 Accès direct aux primitives cryptographiques

7.9.2.1 Généralités

Les UAP peuvent utiliser n'importe lesquels des services cryptographiques disponibles pour un appareil. Ceux-ci comprennent notamment:

- fonctions de hachage non codées et codées;
- génération de chaîne de bits pseudo aléatoire ou aléatoire;
- cryptographie à clés symétriques;
- chiffrement par blocs chiffrants;

NOTE 1 L'exclusion du déchiffrement par blocs chiffrants rend plus probable la capacité d'une mise en œuvre à utiliser l'assistance d'un matériel.

- fonctions de chiffrement continu pour traiter les chaînes de données qui incluent l'authentification, le chiffrement, l'authentification étendue avec chiffrement, déchiffrement, et le déchiffrement avec authentification étendue.

Les primitives cryptographiques disponibles peuvent également inclure une seule option de construction:

- cryptographie à clés asymétriques:
  - chiffrement avec une clé publique et déchiffrement avec une clé privée d'une paire de clés privée/publique;
  - signature avec une clé publique et authentification de signature avec une clé publique d'une paire de clés privée/publique;
  - génération de paires de clés;
  - génération de certificat, signature et autosignature;
  - agrément de clé de Menezes-Qu-Vanstone à deux parties;
  - signatures numériques de Pintsov-Vanstone.

NOTE 2 L'option de construction de cryptographie à clés asymétriques seule fournit l'ensemble de ces capacités.

Les définitions des services abstraits pour l'ensemble de toutes les primitives de 7.9.2 sont spécifiées en 6.2.3.

7.9.2.2 Fonctions de hachage non codées

Une fonction sécurisée de hachage à sens unique non codée (ou à clé fixe) doit être fournie.

Le hachage non codé par défaut doit être celui de Matyas-Meyer-Oseas (MMO) tel que spécifié dans l'ISO/IEC 10118-2, basé sur le chiffrement par blocs de 7.9.3.2.

NOTE L'utilisation de l'algorithme MMO rend plus probable la capacité d'une mise en œuvre à utiliser l'assistance d'un matériel.

D'autres fonctions de hachage non codées peuvent être utilisées lorsque cela est nécessaire, soit en raison de la réglementation nationale, soit parce qu'une plus grande taille de hachage de sortie est exigée pour une certaine application ou pour contrer une menace.

Un paquetage alternatif d'algorithmes cryptographiques peut être exigé pour les systèmes gouvernementaux des USA, parce que la construction MMO n'est pas autorisée pour l'utilisation par le gouvernement des USA. D'autres gouvernements peuvent avoir des politiques semblables.

7.9.2.3 Bits aléatoires

Chaque appareil doit fournir une source de haute qualité de bits aléatoires issus d'un générateur déterministe de bits aléatoires. Celui-ci peut être un générateur à germe correct qui est conforme à l'ANSI X9.82 ou au FIPS 186-3. Lorsqu'elle est disponible, il convient que la source d'entropie élevée soit un générateur non déterministe de bits aléatoires.

Une source de haute qualité de bits aléatoires doit être utilisée dans le rattachement à clés asymétriques. Un générateur déterministe de bits aléatoires à germe correct peut être utilisé pour générer des valeurs de défi dans le rattachement à clés symétriques.

NOTE 1 Les générateurs déterministes de bits aléatoires ne sont pas adaptés pour une utilisation directe en raison de l'incapacité de prouver des propriétés statistiques d'une telle source autres que son non-déterminisme. Au contraire, ils sont utilisés pour semer et fournir une entrée continue à haute entropie à des générateurs déterministes de bits aléatoires, dont les propriétés statistiques sont quantifiables. La certification de la source

d'entropie (comme la certification de la mise en œuvre de sécurité), étant une fonction hautement spécialisée, est mieux déléguée à une entité accréditée. La NIST SP 800-22 est utile pour soumettre à essai les générateurs non déterministes et déterministes de bits aléatoires.

NOTE 2 Dans le processus de rattachement à clés symétriques, il est possible de générer un germe en utilisant le chiffrement par blocs (dont la valeur par défaut est AES) pour chiffrer le temps TAI avec la clé de rattachement (à savoir Seed = Encrypt[K_join, TAI]). Une telle clé de rattachement est présumée provenir d'une source de haute entropie, après avoir été générée dans le gestionnaire de sécurité et distribuée au cours de la phase de configuration.

7.9.3 Cryptographie à clés symétriques

7.9.3.1 Fonctions de hachage codées

Le hachage codé par défaut doit être HMAC, basé sur le hachage non codé de 7.9.2.1 (voir FIPS 198).

7.9.3.2 Fonctions de chiffrement et de déchiffrement par blocs chiffrants

Le chiffrement par blocs par défaut doit être l'AES-128, qui a une taille de bloc B de 16 et une taille de clé de 16 (voir FIPS 197).

Des chiffrements par blocs alternatifs peuvent être utilisés avec l'identificateur d'algorithme approprié lorsque cela est nécessaire, soit en raison de la réglementation nationale, soit parce qu'une plus grande taille de clé ou taille de bloc est exigée pour une certaine application ou pour contrer une certaine menace.

7.9.3.3 Fonctions de chiffrement continu pour le chiffrement, le déchiffrement, l'authentification, l'authentification étendue avec chiffrement et le déchiffrement avec authentification étendue

La sécurité de ce système est basée en partie sur la disponibilité d'un mode de fonctionnement en chiffrement continu d'un chiffrement par blocs qui fournit le chiffrement/déchiffrement, l'authentification, ou les deux. Lorsque tous les deux sont fournis, l'authentification peut s'étendre aux données qui ne sont pas incluses dans le processus de chiffrement/déchiffrement.

NOTE Le chiffrement/déchiffrement sans authentification est évité au sein des TPDU et des DPDU parce qu'il y a un certain nombre d'attaques cryptanalytiques éditées qui s'appliquent à tous les plans de ce type. Cependant, les modes chiffrement seul et déchiffrement seul de CCM* sont disponibles aux UAP pour leur utilisation, tel que pour la protection des données en place.

Le mode de fonctionnement par défaut en chiffrement continu du chiffrement par blocs de 7.9.3.2 doit être CCM* (voir ISO/IEC 19772, mécanisme 3). CCM* peut être utilisé pour l'authentification seule, pour l'authentification étendue avec chiffrement, ou pour le déchiffrement avec authentification étendue.

7.9.3.4 Primitive de génération de clé secrète

Une primitive de génération de clé secrète (SKG) doit être utilisée par les plans d'agrément de clé à clés symétriques spécifiés dans la présente norme.

Cette primitive dérive une valeur secrète partagée à partir d'un défi possédé par une entité U_1 et d'un défi possédé par une entité U_2 lorsque tous les défis partagent les mêmes paramètres de domaine de défi. Si les deux entités exécutent correctement toutes les deux cette primitive avec des défis correspondants comme entrées, la même valeur secrète partagée sera produite.

La valeur secrète partagée doit être calculée comme suit:

- Conditions préalables: les conditions préalables pour l'utilisation de la primitive SKG sont:

  - chaque entité doit être liée à un identificateur unique (par exemple: l'adresse EUI64Address de l'appareil). Tous les identificateurs doivent être des chaînes de bits de la même taille. L'identificateur de l'entité U_1 sera désigné par la chaîne de bits U_1. L'identificateur de l'entité U_2 sera désigné par la chaîne de bits U_2;
  - un plan MAC spécialisé doit avoir été sélectionné, avec les informations d'étiquetage telles que spécifiées dans l'ANSI X9.63:2011, 5.7.1. La taille en bits des clés utilisées par le plan MAC spécialisé est désignée par macKeySize.

- Entrée: la primitive SKG prend comme entrée:
  - une chaîne de bits MacKey de taille macKeySize bits devant être utilisée comme la clé d'un plan MAC spécialisé établi;
  - une chaîne de bits QEU_1 fourni par U_1;
  - une chaîne de bits QEU_2 fourni par U_2.
- Actions: les actions suivantes sont entreprises:
  - former la chaîne de bits constituée de l'identificateur de U_1, de l'identificateur de U_2, la chaîne de bits QEU_1 correspondant au défi de U_1, et la chaîne de bits QEU_2 correspondant au défi de U_2.
- MacData = U_1 || U_2 || QEU_1 || QEU_2.
  - calculer l'étiquette MacTag pour MacData avec la clé MacKey en utilisant la transformation d'étiquetage du plan MAC spécialisé établi:
- MacTag = MACMacKey(MacData)
  - si la transformation d'étiquetage produit ”non valide", produire également ”non valide” et arrêter;
  - autrement, mettre Z=MacTag.
- Sortie: la chaîne de bits Z en tant que valeur secrète partagée.

7.10 Collecte des statistiques de sécurité, détection de menaces et rapports

Il convient que les événements majeurs relatifs à la sécurité consignés dans un journal par le gestionnaire de sécurité incluent:

- les autorisations de nouveaux appareils;
- le premier rattachement de nouveaux appareils au réseau; et
- la disparition prolongée des appareils du réseau, en particulier lorsqu'ils sont censés avoir une présence stationnaire.

NOTE La journalisation exigée d'autres événements de sécurité est un sujet potentiel de normalisation future.

Les événements relatifs à la sécurité suivants doivent tous deux faire l'objet d'une journalisation et d'alertes:

- les taux d'échecs de MIC sur les DPDU reçues qui apparaissent correctement formées spécifiant le correct ID de réseau, qui dépassent une plage spécifiée dans l'attribut 5 du DSMO;
- les taux d'échecs de MIC sur les TPDU reçues qui dépassent une plage spécifiée dans l'attribut 6 du DSMO; et
- tout échec d'intégrité détecté lors du déroulement d'une clé symétrique enveloppée qui excède une plage spécifiée dans l'attribut 9 du DSMO.

7.11 Fonctionnalité de DSMO

7.11.1 Généralités

L'objet de gestion de sécurité d'appareil (DSMO) est une partie intégrante du DMAP et est l'application locale de gestion de sécurité dans chaque appareil. Il est responsable de l'agrément et de l'échange du support cryptographique avec des politiques associées. Il

communique avec le DSMO du gestionnaire de sécurité par l'intermédiaire de l'objet proxy gestionnaire de sécurité (PSMO) du gestionnaire de système. Par conséquent, la sécurité de la TL doit être utilisée pour protéger le trafic du DSMO, sauf pendant le processus de rattachement qui exige des mesures spéciales alternatives.

7.11.2 Attributs du DSMO

Le Tableau 92 décrit le DSMO.

Tableau 92 – Attributs du DSMO

Nom du type d'objet normalisé: Device security management object (DSMO, objet de gestion de sécurité d'appareil)				
Identificateur du type d'objet normalisé: 125				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
DPDU_MIC_Failure_Limit	1	Le seuil d'échec DPDU MIC par unité de temps au-delà duquel une alerte sera envoyée au gestionnaire de sécurité	Type: Unsigned16 Classification: Static Accessibilité: Read/write Valeur par défaut: 5	La valeur est réinitialisée à 0 après la génération d'une alerte
DPDU_MIC_Failure_Time_Unit	2	Temps d'intervalle en secondes utilisé pour déterminer le taux d'échec du MIC DPDU	Type: Unsigned16 Classification: Static Accessibilité: Read/write Valeur par défaut: 60 s	
TPDU_MIC_Failure_Limit	3	Seuil d'échec du MIC DPDU par unité de temps au-delà duquel une alerte sera envoyée au gestionnaire de sécurité	Type: Unsigned16 Classification: Static Accessibilité: Read/write Valeur par défaut: 5 Plage valide: > 0	La valeur est réinitialisée à 0 après la génération d'une alerte
TPDU_MIC_Failure_Time_Unit	4	Temps d'intervalle en secondes utilisé pour déterminer le taux d'échec du MIC DPDU	Type: Unsigned16 Classification: Static Accessibilité: Read/write Valeur par défaut: 5	
DSMO_KEY_Failure_Limit	5	Le seuil au-delà duquel une alerte sera envoyée au gestionnaire de sécurité	Type: Unsigned16 Classification: Static Accessibilité: Read/write Valeur par défaut: 1	La valeur est réinitialisée à 0 après la génération d'une alerte
DSMO_KEY_Failure_Time_Unit	6	Le temps d'intervalle en heures utilisé pour déterminer le taux d'échec clé du DSMO	Type: Unsigned16 Classification: Static Accessibilité: Read/write Valeur par défaut: 1	

Nom du type d'objet normalisé: Device security management object (DSMO, objet de gestion de sécurité d'appareil)				
Identificateur du type d'objet normalisé: 125				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Security_DPDU_Fail_Rate_Exceeded_AlertDescriptor	7	Utilisé pour changer la priorité de l'alerte Security_DPDU_Fail_Rate_Exceeded appartenant à la catégorie de sécurité. Cette alerte peut également être activée ou désactivée	Type: Alert report descriptor Classification: Static Accessibilité: Read/write Valeur par défaut: [false, 6]	Voir la définition d'alerte
Security_TPDU_Fail_Rate_Exceeded_AlertDescriptor	8	Utilisé pour changer la priorité de l'alerte Security_TPDU_Fail_Rate_Exceeded appartenant à la catégorie de sécurité. Cette alerte peut également être activée ou désactivée	Type: Descripteur de rapports d'alertes Classification: Static Accessibilité: Read/write Valeur par défaut: [false, 6]	Voir la définition d'alerte
Security_Key_Update_Fail_Rate_Exceeded_AlertDescriptor	9	Utilisé pour changer la priorité de l'alerte Security_Key_Update_Fail_Rate_Exceeded appartenant à la catégorie de sécurité. Cette alerte peut également être activée ou désactivée	Type: Alert report descriptor Classification: Static Accessibilité: Read/write Valeur par défaut: [false, 6]	Voir la définition d'alerte
pduMaxAge	10	Durée de temps maximale en secondes durant laquelle il est permis pour une PDU de rester sur le réseau. Si une PDU est reçue dans une fenêtre temporelle excédant cette période, il doit être rejeté au récepteur	Type: Unsigned16 Classification: Static Accessibilité: Read/write Valeur par défaut: 510 Plage valide: 0..600	Mis sur 510 s par défaut

7.11.3 KeyDescriptor

7.11.3.1 Généralités

Les informations associées à une clé sont résumées dans le Tableau 93.

Tableau 93 – KeyDescriptor

Nom de l'élément	Identificateur de l'élément	Type de l'élément scalaire
KeyLookupData*	1	Type: OctetString36 Classification: Static Accessibilité: Read/write Voir Tableau 94
KeyUsage	2	Type: Unsigned8 Classification: Static Accessibilité: Read/write Plage valide: 0..7 Voir Tableau 90
ValidNotBefore	3	Type: TAITimeRounded Classification: Static Accessibilité: Read/write
SoftExpirationTime	4	Type: TAITimeRounded Classification: Static Accessibilité: Read/write
ValidNotAfter	5	Type: TAITimeRounded Classification: Static Accessibilité: Read/write
Issuer	6	Type: Adresse IPv6Adress ou EUI64Address Classification: Static Accessibilité: Read/write
CryptoKeyIdentifier	7	Type: Unsigned8 ou Unsigned64 Classification: Static Accessibilité: Read/write
KeyMaterial	8	Type: OctetString Classification: Static Accessibilité: Read/write
SecurityLevel	9	Type: Unsigned8 Classification: Static Accessibilité: Read/write Plage valide: 0..7 Voir Tableau 35
Compteur	10	Type: Unsigned8 Classification: Static Accessibilité: Read/write
NonceCache	11	Type: OctetString Classification: Dynamic Accessibilité: Read/write

Nom de l'élément	Identificateur de l'élément	Type de l'élément scalaire
MICFailures	13	Type: Unsigned16 Classification: Static Accessibilité: Read/write
NOTE * indique un champ d'indice.		

Les champs T-keyLookupData OctetString sont énumérés dans le Tableau 94.

Tableau 94 – Champs T-keyLookupData OctetString

Nom de champ	Type du champ scalaire
SourceAddress	Type: IPv6Address
SourcePort	Type: Unsigned16
DestinationAddress	Type: IPv6Address
DestinationPort	Type: Unsigned16

NOTE 1 Puisque la représentation interne du Key Descriptor n'est pas observable, tous les éventuels aspects de représentation dans ce qui suit sont purement donnés pour l'exposé.

Les champs Key Descriptor incluent:

- KeyLookupData:
  - à la TL, utilisé comme indice pour trouver une clé pour une association donnée;
  - à la DL, ce champ n'est pas utilisé et doit être mis à 0x00 partout;
- KeyUsage: permet d'identifier si la clé est utilisable en tant que clé D, clé T ou les deux;

NOTE 2 L'utilisation d'une carte de bits à 2 bits permet à une clé d'être définie pour être utilisée comme clé de DL et clé de T en même temps, si cela est permis par la politique de clé.

- ValidNotBefore: temps (TAI) auquel la clé devient valide;
- SoftExpirationTime: temps (TAI) auquel une clé mise à jour est nécessaire;
- ValidNotAfter: temps (TAI) auquel la clé devient non valide;
- Issuer: adresse de l'émetteur de la clé; cela peut être une adresse IPv6Adress ou une adresse EUI64Address;
- CryptoKeyIdentifier: CryptoKeyIdentifier, mis par l'émetteur de clé, utilisé pour distinguer des clés lorsque plusieurs clés sont valides simultanément;
- KeyMaterial: données de clé pour chiffrement/déchiffrement et/ou génération de MIC;
- SecurityLevel: similaire à ce qui est décrit en Tableau 88;
- Counter: si le KeyUsage bit0 est 0 (cette clé n'est pas une clé D), ce champ n'est pas utilisé, donc il est mis à 0;
- Noncecache: si le KeyUsage bit1 est 0 (cette clé n'est pas une clé T), ce champ n'est pas utilisé, donc il est mis à NULL;
- MICFailure: nombre d'échecs d'authentification de MIC après lequel il convient de générer une alarme.

7.11.3.2 Méthodes complémentaires d'objet de gestion de sécurité d'appareil pour prendre en charge la gestion de clé

Le Tableau 95 décrit la méthode de suppression de clé. Le résultat de l'invocation de méthode est stocké dans ServiceFeedbackCode dans l'en-tête de sous-couche d'application

et retourné à l'appareil demandeur. La construction de nonce pour protéger l'APDU utilisant la clé principale est décrite en 7.5.3.

Tableau 95 – Méthode Delete_key

<table>
<tr><th colspan="5">Nom du type d'objet normalisé: Device security management object (DSMO, objet de gestion de sécurité d'appareil)</th></tr>
<tr><th colspan="5">Identificateur du type d'objet normalisé: 125</th></tr>
<tr><th>Nom de la méthode</th><th>ID de la méthode</th><th colspan="3">Description de la méthode:</th></tr>
<tr><td rowspan="12">Delete_key</td><td>2</td><td colspan="3">Cette méthode est utilisée pour effacer une clé symétrique d'un appareil. Cette méthode est évoquée par le PSMO du gestionnaire de sécurité. Cette méthode doit être protégée par la clé principale courante partagée par l'appareil et le gestionnaire de sécurité</td></tr>
<tr><td></td><td colspan="3">Arguments d'entrée</td></tr>
<tr><td></td><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td></td><td>1</td><td>KeyUsage</td><td>Unsigned8</td><td>KeyUsage défini dans le Tableau 90</td></tr>
<tr><td></td><td>2</td><td>Crypto_Key_Identifier</td><td>Unsigned8</td><td>Le Crypto_Key_Identifier utilisé pour identifier de façon unique les clés superposées au cours de la période de validité</td></tr>
<tr><td></td><td>3</td><td>Source_Port</td><td>Unsigned16</td><td>Port source; si KeyUsage n'est pas 0x01 (une clé T par exemple), il convient d'élider ce champ</td></tr>
<tr><td></td><td>4</td><td>Destination_Address</td><td>IPv6Adress</td><td>Adresse de destination; si KeyUsage n'est pas 0x01 (une clé T par exemple), il convient d'élider ce champ</td></tr>
<tr><td></td><td>5</td><td>Destination_Port</td><td>Unsigned16</td><td>Port de destination; si KeyUsage n'est pas 0x01 (une clé T par exemple), il convient d'élider ce champ</td></tr>
<tr><td></td><td>6</td><td>MasterKeyID</td><td>Unsigned8</td><td>CryptoKeyIdentifier pour la clé principale utilisée pour la génération de MIC</td></tr>
<tr><td></td><td>7</td><td>Time_Stamp</td><td>Unsigned32</td><td>Temps de création de ce message sous la forme TAITimeRounded. Cet argument est la portion de temps du nonce utilisé pour la génération de MIC pour la protection de cet appel de méthode</td></tr>
<tr><td></td><td>8</td><td>MIC</td><td>OctetString (SIZE = 4, 8, 16)</td><td>La vérification de l'intégrité utilisant AES_CCM*. La taille de MIC est sélectionnée dans MIC-32, MIC-64 et MIC-128 avec le niveau de sécurité de clé principale assigné dans le processus de rattachement</td></tr>
<tr><td></td><td colspan="3">Arguments de sortie</td></tr>
<tr><td></td><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td></td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
</table>

Le MIC est généré par une opération de l'AES-CCM* avec les paramètres suivants:

- partie authentification: élément 1..7;
- partie cryptage: aucune;
- clé: clé principale, qui a CryptoKeyIdentifier = MasterKeyID;
- nonce: formé avec la structure du Tableau 57 avec:
  - EUI64Address: Adresse EUI64Address du gestionnaire de sécurité;
  - temps TAI nominal: champ Time Stamp acheminé dans la demande de Delete_Key().

La méthode Key_Policy_Update est décrite dans le Tableau 96. Le résultat de l'invocation de méthode est stocké dans ServiceFeedbackCode dans l'en-tête de sous-couche d'application et retourné à l'appareil demandeur. La construction de nonce pour protéger l'APDU utilisant la clé principale est décrite en 7.5.3.

Tableau 96 – Méthode Key_Policy_Update

Nom du type d'objet normalisé: Device security management object (DSMO, objet de gestion de sécurité d'appareil)				
Identificateur du type d'objet normalisé: 125				
Nom de la méthode	**ID de la méthode**	**Description de la méthode**		
Key_Policy_Update	3	Cette méthode est utilisée pour mettre à jour une politique associée à une clé symétrique sur un appareil. Cette méthode est évoquée par le PSMO du gestionnaire de sécurité. Cette méthode doit être protégée par la clé principale courante partagée par l'appareil et le gestionnaire de sécurité		
	Arguments d'entrée			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	KeyUsage	Unsigned8	KeyUsage défini dans le Tableau 90
	2	Crypto_Key_Identifier	Unsigned8	Le Crypto_Key_Identifier utilisé pour identifier de façon unique les clés superposées au cours de la période de validité
	3	Source_Port	Unsigned16	Port source; si KeyUsage n'est pas 0x01 (une clé T par exemple), il convient d'élider ce champ
	4	Destination_Address	IPv6Adress	Adresse de destination; si KeyUsage n'est pas 0x01 (une clé T par exemple), il convient d'élider ce champ
	5	Destination_Port	Unsigned16	Port de destination; si KeyUsage n'est pas 0x01 (une clé T par exemple), il convient d'élider ce champ
	6	SoftLifeSpanRatio	Unsigned8	Le pourcentage de HardLifeSpan à partir duquel une mise à jour de la clé sera initiée
	7	Security_Level	Unsigned8	Security Level (niveau de sécurité) spécifié dans le Tableau 35
	8	MasterKeyID	Unsigned8	CryptoKeyIdentifier pour la clé principale utilisée pour la génération de MIC

Nom du type d'objet normalisé: Device security management object (DSMO, objet de gestion de sécurité d'appareil)				
Identificateur du type d'objet normalisé: 125				
	9	Time_Stamp	Unsigned32	Temps de création de ce message sous la forme TAITimeRounded. Cet argument est la portion de temps du nonce utilisé pour la génération de MIC pour le cryptage et la protection de cet appel de méthode
	10	MIC	OctetString (SIZE = 4, 8, 16)	La vérification de l'intégrité utilisant AES_CCM*. La taille de MIC est sélectionnée dans MIC-32, MIC-64 et MIC-128 avec le niveau de sécurité de clé principale assigné dans le processus de rattachement
	Arguments de sortie			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	-	-	-	-

Le MIC est généré par une opération de l'AES-CCM* avec les paramètres suivants:

- partie authentification: élément 1..7;
- partie cryptage: aucune;
- clé: clé principale, qui a CryptoKeyIdentifier = MasterKeyID;
- nonce: formé avec la structure du Tableau 57 avec:
  - EUI64Address: Adresse EUI64Address du gestionnaire de sécurité;
  - temps TAI nominal: Champ Time Stamp acheminé dans la demande de Key_Policy_Update().

Le SoftExpirationTime dans le Key Descriptor est mis à jour après un contrôle de MIC réussi sur les paramètres de cet appel de méthode. Tous les paramètres doivent être concaténés du premier élément à celui situé avant le dernier élément (excluant ainsi la vérification d'intégrité). La SoftLifeSpanRatio de clé est le pourcentage de la différence entre les temps ValidNotAfter et ValidNotBefore. Par exemple, une SoftLifeSpanRatio de 50 % causerait une mise à jour de clé à mi-chemin entre la ValidNotBefore et la ValidNotAfter.

7.11.4 Alertes de DSMO

Le Tableau 97 décrit les alertes de DSMO.

Tableau 97 – Alertes de DSMO

Nom du type d'objet normalisé: Device security management object (DSMO, objet de gestion de sécurité d'appareil)					
Identificateur du type d'objet normalisé: 125					
Description de l'alerte: Alertes de sécurité sur l'état de la communication					
Classe d'alertes (Enumerated: alarme ou événement)	**Catégorie d'alertes (Enumerated: diagnostic d'appareil, diagnostic de comm., sécurité ou processus)**	**Type d'alertes (Enumerated: en fonction de la catégorie d'alertes)**	**Priorité d'alertes (Enumerated: urgent, haut, moyen, faible, journal)**	**Type de données de la valeur**	**Description de la valeur incluse avec l'alerte**
0 = événement	2 = sécurité	0 = Security_DPDU_Fail_Rate_Exceeded	6 = Moyen	Type: Unsigned16 Valeur par défaut: 0	Alerte générée après le dépassement du seuil du taux d'échec DPDU préconfiguré. La valeur indique le nombre d'échecs au cours de la période de temps
0 = événement	2 = sécurité	1 = Security_TPDU_Fail_Rate_Exceeded	6 = Moyen	Type: Unsigned16 Valeur par défaut: 0	Alerte générée après le dépassement du seuil du taux d'échec TPDU préconfiguré. La valeur indique le nombre d'échecs au cours de la période de temps
0 = événement	2 = sécurité	2 = Security_Key_Update_Fail_Rate_Exceeded	6 = Moyen	Type: Unsigned16 Valeur par défaut: 0	Alerte générée après le dépassement du seuil du taux d'échec de mise à jour de clé de sécurité préconfiguré. La valeur indique le nombre d'échecs au cours de la période de temps

8 Couche physique

8.1 Généralités

La couche physique (PhL) est responsable de convertir les informations de données numériques vers, et à partir de, l'énergie radioélectrique émise, et capturée, par une antenne d'appareil. L'Article 8 spécifie également les fréquences opérationnelles, les niveaux de puissance d'émission et les méthodes de modulation utilisées. Comme décrit en 5.2.6.2, la présente norme utilise l'étalement du spectre en séquence directe DSSS dans la bande de 2,4 GHz selon l'IEEE 802.15.4 comme la couche PhL par défaut, à laquelle elle se réfère comme le moyen de terrain de Type A. (voir 5.2.6.4). Les futures versions de la présente norme peuvent définir des couches physiques alternatives.

La PhL fournit deux services, le service de données de PhL et le service de gestion de PhL. Ces services sont collectivement accessibles par l'intermédiaire du PhSAP. Le service de données de PhL (PhD) permet l'émission et la réception des données utilisateur réelles (les PhPDU) à travers le canal radio physique. Le service de gestion de PhL est utilisé pour

commander les fonctions opérationnelles de la radio telles que le choix de canal, la sélection de la puissance d'émission, etc.

La structure des PhPDU utilisées par la présente norme est définie dans IEEE 802.15.4. Chaque PhPDU est constituée d'un en-tête de synchronisation PHY (PSH), d'un en-tête de codage PHY (PCH), et d'une charge utile de PHY qui est une PhSDU unique. Un délimiteur de début de trame (SFD) dans le SHR est communément utilisé comme référence de temporisation observable.

Un appareil employant une radio DSSS dans la bande de 2,4 GHz certifiée conforme à l'IEEE 802.15.4, sera généralement autorisé à fonctionner sans licence de site dans la plupart des pays dans le monde. Un contrat de licence de l'appareil acceptable pour le(s) pays dans le(s)quel(s) l'appareil est destiné à être utilisé peut être exigé.

8.2 Couche physique par défaut

8.2.1 Exigences générales

La couche physique par défaut doit être la PhL de Type A (voir 5.2.6.4), qui doit être basée sur DSSS dans la bande de 2,4 GHz selon l'IEEE 802.15.4 avec des exigences et des exceptions complémentaires telles que spécifiées dans le présent document.

Les appareils sur les plates-formes mobiles telles que des navires et des trains et même des camions peuvent se déplacer entre différentes juridictions réglementaires. Dans tous les cas, les réglementations pour le lieu courant de l'appareil s'appliquent. Un des paramètres de configuration d'appareil, dlmo.CountryCode (9.1.15.6), fournit les conseils relatifs au lieu et aux contraintes de réglementation exigées pour piloter la conformité aux réglementations pertinentes; ainsi, pour les plateformes mobiles, la valeur de ce paramètre doit être changée au cours de l'exploitation du système, en temps utile, selon les besoins, pour être conforme aux réglementations pertinentes. Voir 5.2.5.3 et l'Annexe V.

Le fournisseur d'appareil et l'utilisateur final sont responsables de certifier que ces appareils sont conformes à la présente norme et à toutes les éventuelles réglementations spécifiques à un pays ou à une région.

8.2.2 Exigences complémentaires de l'IEEE 802.15.4

8.2.2.1 Débit de données par liaison radio

La PhL doit prendre en charge un débit de données brut (par liaison radio) de 250 kbit/s.

8.2.2.2 Exigences relatives à la temporisation

La présente norme exige que la PhLE prenne en charge le changement de canal pour chaque PhPDU émise. Les exigences relatives à la temporisation sont spécifiées dans le Tableau 98.

Tableau 98 – Exigences relatives à la temporisation

Evénement	Exigence
Temps de changement des voies RF	< 200 µs
Temps de commutation de réception à émission (avec la PA activée)	< 200 µs
Temps de commutation d'émission (avec la PA activée) à réception	< 200 µs
Temps de préparation d'inter-réception	< 200 µs
NOTE PA fait référence à tout amplificateur de puissance RF dans l'appareil.	

8.2.2.3 Détection de porteuse sélection du mode

La couche physique DSSS 2,4 GHz de l'IEEE 802.15.4 prend en charge l'utilisation d'un plan CSMA/CA pour réduire les collisions et augmenter la coexistence. Ce plan peut retarder de façon excessive l'émission d'une PhPDU, en raison de retards de replis aléatoires répétitifs pendant l'acquisition de canal.

La PhLE doit sélectionner le mode de fonctionnement CSMA/CA sur une base de transaction D à transaction D conformément à la demande faite par la DLE, selon la configuration de système, y compris le régime de réglementation dans lequel le système sans fil fonctionne, tel que contraint par dlmo.CountryCode (9.1.15.6).

8.2.2.4 Nombre de canaux

La PhLE doit prendre en charge à tout le moins les canaux 11..25 de DSSS 2,4 GHz de l'IEEE 802.15.4. La prise en charge du canal 26 de l'IEEE 802.15.4 est facultative lorsque son utilisation est permise par des contraintes réglementaires, et interdite lorsque les réglementations interdisent son utilisation.

NOTE L'utilisation du canal 26 est facultative en raison des contraintes réglementaires communément rencontrées près de la frontière de bande.

8.2.2.5 Limites de puissance d'émission

Comme spécifié par IEEE 802.15.4, la PhLE doit prendre en charge un niveau minimal de pleine puissance de -3 dBm, mesuré conformément aux réglementations par rapport auxquelles l'appareil est certifié.

La PhLE doit fournir une puissance d'émission réglable de -5 dBm à la puissance maximale de l'appareil, en des incréments spécifiés par l'attribut TXPowerTolerance de la PhE, tel que spécifié dans l'IEEE 802.15.4 PHY PIB.

Conformément aux exigences de l'IEEE 802.15.4:2011, 8.1.5, le niveau maximal de puissance rayonnée ne doit pas dépasser les exigences réglementaires qui s'appliquent à l'endroit où l'appareil est déployé, comme contraint par dlmo.CountryCode (9.1.15.6).

8.2.3 Exceptions à la couche physique de l'IEEE 802.15.4

8.2.3.1 Généralités

Les exigences de la présente norme qui sont des écarts ou des omissions par rapport à la couche physique de l'IEEE 802.15.4 sont énumérées ici.

8.2.3.2 Limitation de bandes de fréquence et classes de modulation

Bien que la couche physique de l'IEEE 802.15.4 prenne en charge plusieurs bandes de fréquence et classes de modulation, un appareil conforme à la présente norme doit fonctionner dans la bande de 2 400..2 483,5 MHz sans licence utilisant la modulation DSSS (et le codage à 250 kbit/s), qui est spécifiée dans l'IEEE 802.15.4:2011, Tableau 66, comme étant DSSS 2 450. La présente norme ne prend en charge aucune des autres bandes de fréquences, aucun des autres débits de données et aucune des autres techniques de modulation et de codage spécifié(e)s dans l'IEEE 802.15.4.

9 Couche liaison de données

9.1 Généralités

9.1.1 Vue d'ensemble

La couche liaison de données (DL) dans la présente norme est conçue dans le but général de contraindre la gamme des options reconnues de construction pour un appareil de terrain, tout en permettant des solutions de système flexibles et innovantes.

La spécification de DL fournit un ensemble de capacités qui sont bien définies et vérifiables pour chaque appareil qui participe à un sous-réseau D. La DLE peut être conceptualisée comme étant un diagramme d'état à base de tableaux qui fonctionne indépendamment sur chaque appareil. Un sous-réseau D est un groupe de DLE équipées d'un jeu assorti de configurations à base de tableaux par le gestionnaire de système.

Les blocs modules de base de la DL comprennent des intervalles de temps, des supertrames, des liaisons et des graphes. Le gestionnaire de système peut assembler ces blocs modules de base pour configurer une DLE en l'une de trois variantes opérationnelles générales, à savoir le saut de voie discrétisé (slotted-channel-hopping), le saut de voie lent (slow-channel-hopping), et le saut de voie hybride discrétisé/lent (hybrid slotted/slow-channel-hopping). (Voir 9.1.7.2 pour un débat relatif au saut de voie.)

Un intervalle de temps est une seule période de temps non répétitive. Les durées d'intervalle de temps dans la présente norme sont configurables à une valeur fixe telle que 10 ms ou 12 ms. Une fois qu'une durée d'intervalle de temps est choisie, tous les intervalles de temps ont généralement la même durée et ils sont réalignés sur un cycle de 4 Hz à chaque intervalle d'horloge de 250 ms. (Voir 9.1.9 pour un débat relatif au chronométrage de la DLE.)

Une supertrame est une collection d'intervalles de temps se répétant sur un programme cyclique. Le nombre d'intervalles de temps dans une supertrame donnée détermine la fréquence de répétition de chaque intervalle de temps, établissant ainsi un cycle de communication pour les DLE qui utilisent la supertrame. La supertrame a également un modèle de saut de voie de référence qui lui est associé. (Voir 9.1.8 pour un débat relatif aux supertrames.)

Les liaisons sont des connexions entre des DLE. Lorsque le gestionnaire de système définit des chemins entre des DLE, les DLE reçoivent des attributions de liaison. Une attribution de liaison se répète sur un programme cyclique, par le biais de sa connexion à une supertrame sous-jacente. Chaque liaison se réfère à un intervalle de temps ou à un groupe d'intervalles de temps au sein d'une supertrame, à son type (émission et/ou réception), à des informations relatives au voisin de la DLE (la DLE sur l'autre extrémité de la liaison), à un décalage de voie par rapport au modèle de saut de voie sous-jacent de la supertrame, et à des alternatives d'émission/réception.

La présente norme prend en charge le routage par graphe ainsi que le routage de source. Un graphe orienté est un ensemble de liaisons orientées qui est utilisé pour acheminer des DPDU Data au sein d'un sous-réseau D. Chaque graphe orienté au sein du sous-réseau D est identifié par un ID de graphe. Dans le routage de source, la DLE émettrice désigne le chemin saut par saut que la DPDU Data emprunte à travers un sous-réseau D. Le routage par graphe et le routage par source peuvent être mélangés. (Voir 9.1.6 pour un débat relatif au routage.)

9.1.2 Stratégies de coexistence dans la DL

La présente norme incorpore plusieurs stratégies qui sont utilisées simultanément pour optimiser la coexistence avec d'autres utilisateurs du spectre radio de 2,4 GHz, tel que décrit en 4.6.10. La plupart de ces stratégies sont traitées de manière adaptative par la DLE conjointement avec le gestionnaire de système.

9.1.3 Allocation de largeur de bande numérique

La DLE est un diagramme d'états à base de tableaux qui fournit l'accès hiérarchisé par ordre de priorité à la largeur de bande numérique pour la communication directionnelle parmi des DLE au sein d'un sous-réseau D. La machine à états fonctionne sur un seul intervalle de temps à la fois.

La largeur de bande numérique est allouée par une fonction de gestion de système. Par exemple, un appareil de terrain peut avoir besoin de faire rapport toutes les 10 s. Un niveau de service est configuré par le biais de la fonction de gestion de système, pour assurer que la largeur de bande numérique est disponible en cas de besoin. La fonction de gestion de système, à son tour, configure la largeur de bande numérique disponible pour la DLE de l'appareil de terrain et de toutes les éventuelles DLE de routeurs intermédiaires exigées.

Une liaison est l'unité de base de service au sein de la DL. Une liaison peut être entrante, sortante, ou bidirectionnelle. Elle peut être en monodiffusion ou en diffusion (voir 9.1.9.4.2).

La largeur de bande numérique de DL peut être allouée pour délivrer un niveau moyen de service, par exemple, 10 DPDU Data de largeur de bande numérique disponible (multi-saut) par minute. En variante, la largeur de bande numérique de DL peut être allouée pour prendre en charge un intervalle de production de rapports et un niveau de service, par exemple, une DPDU Data (y compris les répétitions de tentatives) toutes les 15 s, avec une latence maximale de 2 s à une connexion dorsale.

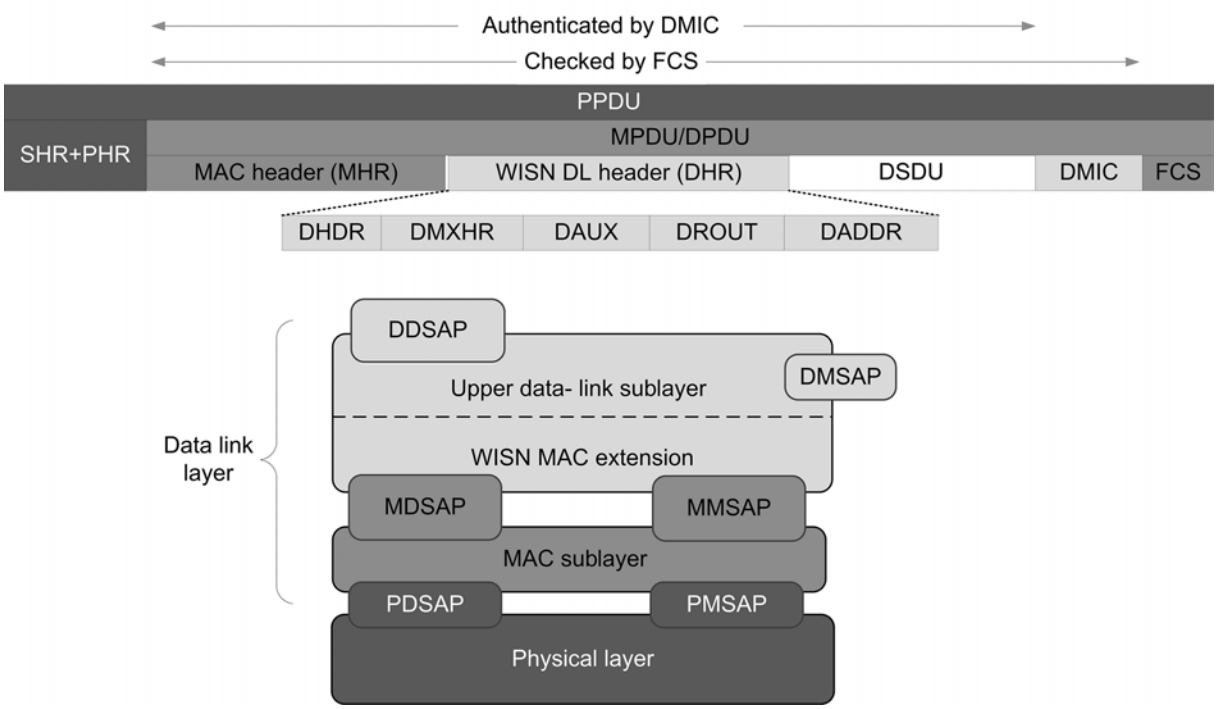
La largeur de bande numérique de DL peut être organisée comme un pot commun qui peut être partagé par un ensemble de DLE utilisant les liaisons correspondantes. Un niveau de service peut être fourni en assurant qu'une capacité partagée en mode contention suffisante est disponible.

Pour une allocation de voie plus granulaire, les liaisons peuvent être attachées à des groupes particuliers de DPDU Data.

Un niveau de service peut être fourni avec une combinaison d'allocations spécifiques de liaison et de largeur de bande numérique partagée généralement disponible. Par exemple, pour chaque rapport, une liaison spécialisée peut être allouée pour la première émission à chacun de deux voisins, avec des répétitions de tentative utilisant la largeur de bande numérique partagée.

9.1.4 Structure de la DPDU

La structure générale d'une unité de données de protocole de liaison de données (DPDU) dans la présente norme est montrée à la Figure 53.



Légende

Anglais	Français
Authenticated by DMIC	Authentifié par DMIC
Checked by FCS	Vérifié par FCS
MAC Header (MHR)	En-tête MAC (MHR)
WISN DL header (DHR)	En-tête WISN DL (DHR)
Data link layer	Couche liaison de données
Upper data-link sub-layer	Sous-couche liaison de données supérieure
WISN MAC extension	Extension WISN MAC
MAC sub-layer	Sous-couche MAC
Physical layer	Sous-couche physique

Figure 53 – Suite de protocoles de DL et structure de PhPDU/DPDU

La DL spécifiée par la présente norme comprend:

- Un sous-ensemble de la couche MAC de l'IEEE 802.15.4, telle que décrite en 9.1.5. Celui-ci gère les mécanismes de bas niveau d'envoi et de réception de DPDU individuelles (dont toutes sont classifiées comme DPDU Data par l'IEEE 802.15.4). Les SHR, PHR, MHR, et séquence de contrôle de trame (FCS) de chaque DPDU sont tels que décrits et spécifiés dans l'IEEE 802.15.4.
- Une extension de la MAC, y compris les aspects de la DL qui ne sont pas spécifiés par l'IEEE mais sont logiquement des fonctions MAC.
- Un protocole de DL supérieur qui traite les aspects liaison et maille au-dessus du niveau MAC.

Des composants de l'en-tête de DPDU dans la présente norme sont décrits en 9.3.1.

9.1.5 La DL et la MAC de l'IEEE 802.15.4

La présente norme utilise la couche MAC de l'IEEE 802.15.4 (appelée IEEE MAC dans le présent document). Seules les trames de données de l'IEEE MAC sont utilisées par la DL. Les formats utilisés sont tels que spécifiés par l'IEEE 802.15.4. Deux exceptions sont explicitement énumérées en 9.1.5. Voir 9.3.3 pour plus d'informations.

L'IEEE MAC décrit diverses caractéristiques qui ne sont pas utilisées par la DL de la présente norme (appelée "DL" dans le présent document). En résumé, seules les trames de données de l'IEEE MAC sont utilisées par la DL.

Une DLE conforme à la présente norme ne s'associe jamais à un coordonnateur dans le sens défini par l'IEEE MAC. Aucune des fonctions de l'IEEE MAC impliquant des appareils pleines fonctions (FFD[8]) n'est utilisée par la présente norme.

Dans le contexte limité des DPDU de la présente norme, il y a quelques caractéristiques qui ne sont pas prises en charge pour les trames de données IEEE MAC. Ces caractéristiques sont mises en œuvre par l'intermédiaire de l'extension MAC de la présente norme, qui renforce l'IEEE MAC avec des caractéristiques qui sont logiquement des fonctions de MAC, mais qui ne sont pas incluses dans l'IEEE 802.15.4.

La DL et le MAC de l'IEEE 802.15.4 spécifient chacune une entité appelée une supertrame (voir 9.1.8), mais la DL n'utilise aucun aspect de la spécification de supertrames de l'IEEE 802.15.4 MAC.

Des DPDU ACK/NAK sont utilisées pour acheminer des informations relatives au temps pour une correction d'horloge, en plus de fournir un acquittement authentifié. Ces caractéristiques ne sont pas disponibles lors de l'utilisation d'une immédiate MPDU d'acquittement de l'IEEE 802.15.4 MAC. Pour cette raison, entre autres, les immédiates d'acquittement de niveau MAC spécifiées dans l' IEEE 802.15.4 ne sont pas utilisées. A la place, des acquittements immédiats de niveau MAC conformes à la présente norme sont ici fournis, tels que des trames de données IEEE 802.15.4 courtes qui utilisent habituellement une combinaison de structure de champ d'adresse non destinée à ces fins dans l'IEEE 802.15.4.

NOTE Les DPDU ACK/NAK sont des SCS (Short Control Signaling), tels que spécifiés dans l'ETSI EN 300 328.

L'IEEE 802.15.4 MAC inclut des balayages actifs et passifs, qui ne sont pas utilisés dans la présente norme. La présente norme a des balayages actifs et passifs alternatifs, utilisant des trames de données de l'IEEE 802.15.4.

Le mécanisme de repli et de répétition de tentative de l'IEEE 802.15.4 MAC n'est pas utilisé par la DL. Au lieu de cela, la DL met en œuvre ses propres répétitions de tentative, impliquant la diversité spatiale (répétitions de tentative à plusieurs DLE), la diversité de fréquence (répétitions de tentative sur plusieurs canaux radio), et la diversité de temps (retardant la DPDU Data). La façon et le degré de ces éléments de diversité ne sont pas fixes, mais ils sont configurés par le gestionnaire de système. Plus généralement, la DL de la présente norme utilise l'accès CSMA/CA, mais les détails sont différents de l'utilisation du CSMA/CA tel que défini dans l'IEEE 802.15.4. Divers aspects du comportement en CSMA/CA de l'IEEE 802.15.4 MAC ne sont pas utilisés, et les fonctions de CSMA/CA sont traitées dans la DL de la présente norme.

La présente norme comprend deux exceptions aux combinaisons d'adressage MAC-PDU spécifiées dans l'IEEE 802.15.4:

- Les DPDU Data de sollicitation et la plupart des DPDU ACK/NAK, qui sont techniquement des trames de données dans l'IEEE 802.15.4, utilisent un mode d'adressage de destination de 00 et un mode d'adressage de source de 00. Dans l'IEEE 802.15.4, cette combinaison se limite aux balises de l'IEEE 802.15.4 et aux immédiates d'acquittement trivialement brouillées de l'IEEE 802.15.4.
- Les DPDU Data d'annonce et les DPDU ACK/NAK secondaires en duodiffusion/N-diffusion qui sont techniquement des trames de données dans l'IEEE 802.15.4, utilisent un mode d'adressage de destination de 00 et un mode d'adressage de source de 10 (adresse DL16Address). Dans l'IEEE 802.15.4, cette combinaison implique que

[8] FFD = *full function device*.

la trame est dirigée vers le coordonnateur de réseau dans un local (PAN), ce qui n'existe pas dans la présente norme (et qui, par conséquent, ne s'applique pas).

9.1.6 Chemins et graphes

9.1.6.1 Généralités

Des chemins sont configurés par le gestionnaire de système, en se basant sur des rapports issus des DLE qui spécifient la qualité instantanée et historique de la connectivité sans fil vers leurs voisins immédiats. Le gestionnaire de système accumule ces rapports de qualité de signal pour prendre des décisions de routage. Les rapports de qualité de signal sont normalisés, mais le processus de décision de routage au sein du gestionnaire de système n'est pas normalisé. Une fois que le gestionnaire de système prend ses décisions de routage, il utilise des DPDU Data normalisées pour configurer les chemins au sein de chaque DLE dans le sous-réseau D. (Voir 9.1.13 et 9.1.14 pour une revue de la découverte de voisins).

Le routage de DL est adaptatif en deux niveaux:

- Les DLE prennent des décisions de transmission adaptatives instantanées. Les DLE sont normalement configurées avec la diversité de chemin et, de ce fait, si une liaison échoue quelque part le long du chemin, la DLE peut immédiatement envoyer la DPDU Data le long d'un chemin alternatif.
- Si, au fil du temps, certaines liaisons ont de constants problèmes de connectivité, cela est rapporté au gestionnaire de système, qui peut alors reconfigurer la DLE pour utiliser des liaisons différentes.

Dans chaque DPDU Data, des instructions de routage de DL sont placées dans le sous-en-tête DROUT de la DPDU Data (voir 9.3.3.6). Lorsque des DPDU Data sont adressées à un voisin immédiat, comme au cours du processus de rattachement au sous-réseau D, le chemin est simplement l'adresse du voisin en question. Lorsque les DPDU Data sont envoyées à une DLE plus distante, un simple numéro de graphe peut spécifier comment il convient d'acheminer la charge utile de la DPDU Data à travers le sous-réseau D pour atteindre l'adresse en question. Ces deux approches peuvent être combinées. Par exemple, un sous-en-tête DROUT peut contenir deux entrées, la première identifiant un voisin immédiat pour le premier saut, et la seconde indiquant un graphe utilisé pour le reste du chemin à travers le sous-réseau D.

Les chemins qui identifient des adresses D spécifiques, également connus comme routage de source, ne sont pas aussi adaptatifs que les chemins basés sur des graphes. Lorsqu'un chemin est basé sur une série d'adresses D, chaque DLE le long du chemin devient un point unique d'échec. D'autre part, il convient que les graphes soient configurés avec plusieurs branches à chaque saut, afin qu'en cas de problème de connectivité avec un voisin, la DLE puisse envoyer la charge utile de cette DPDU Data vers un voisin différent.

Le routage de source est utile pour des communications rapides et transitoires entre des DLE, comme au cours du processus de rattachement. Le routage de source peut également être utilisé lorsque les ressources des chemins de graphe sont rares.

Le sous-en-tête DROUT est construit par la DLE qui injecte la DPDU Data dans la DL à partir d'une NLE. Les chemins sont sélectionnés par une consultation de table basée sur l'ID de contrat, l'adresse D de destination D, ou par défaut. Une fois un chemin sélectionné, la charge utile acheminée de la DPDU Data suit ce chemin jusqu'à ce qu'elle arrive au point de terminaison du sous-réseau D, qui peut être sa destination finale, ou alternativement peut être un point de cheminement le long du chemin, tel qu'un routeur dorsal ou le gestionnaire de système. Au point de terminaison du sous-réseau D, la charge utile de la DPDU Data reçue est passée à la NLE contiguë.

Le sous-en-tête DROUT inclut un champ de limite de transmission qui est utilisé pour limiter le nombre de fois qu'une DPDU peut être transmise au sein d'un sous-réseau D. La limite de transmission est initialisée par la DLE initiatrice lorsque le chemin est assigné, et

décrémentée avec chaque saut jusqu'à ce qu'elle atteigne zéro, déclenchant le rejet de la DPDU Data si un saut ultérieur était exigé.

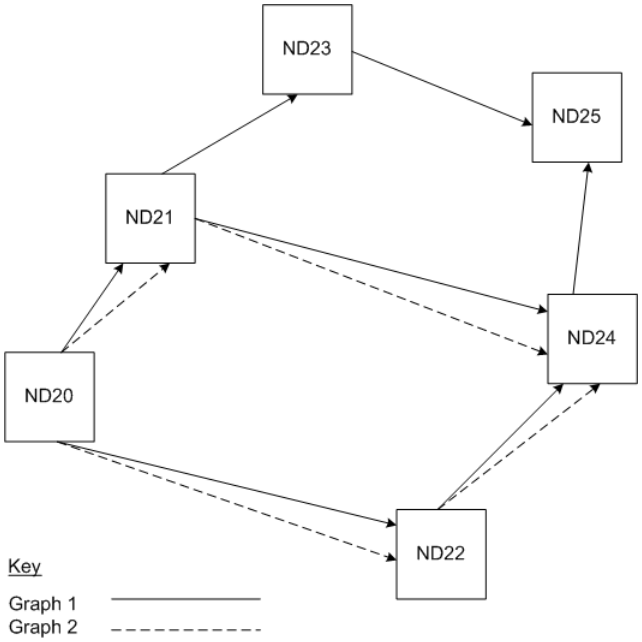
NOTE Cette limite de transmission s'assure que les DPDU Data ne peuvent pas circuler pour toujours par un chemin circulaire non intentionnel.

9.1.6.2 Routage par graphe

Un graphe est un ensemble de liaisons orientées qui est utilisé pour acheminer des messages au sein d'un sous-réseau D. Chaque graphe désigné par le gestionnaire de système pour le routage au sein d'un sous-réseau D est identifié par un identificateur de graphe (graph ID).

Les liaisons associées à chaque graphe sont configurées par le gestionnaire de système. Un sous-réseau D peut avoir plusieurs graphes, dont certains peuvent se chevaucher. Chaque DLE peut avoir plusieurs graphes passant par elle, voire vers les mêmes voisins.

La Figure 54 montre un exemple de routage par graphe.



Légende

Anglais	Français
Key	Légende
Graph 1	Graphe 1
Graph 2	Graphe 2

Figure 54 – Exemple de routage par graphe

Sur la Figure 54, ND20 communique avec ND25 en utilisant le graphe 1. Pour envoyer la charge utile d'une DPDU Data sur ce graphe, ND20 peut la transmettre à ND21 ou à ND22. A partir de ces DLE, la charge utile de la DPDU Data peut prendre plusieurs chemins alternés, mais toutefois, en suivant le graphe 1, la charge utile de la DPDU Data arrivera à ND25. De même, pour communiquer avec ND24, ND20 peut envoyer des DPDU Data sur le graphe 2 par ND21 ou ND22, dont l'un ou l'autre, à son tour, transmettra la charge utile de la DPDU Data à ND24.

La Figure 54 montre tous les graphes provenant de ND20, mais les mêmes graphes peuvent être utilisés par n'importe quel nœud. Par exemple, le gestionnaire de système peut configurer ND21 pour utiliser le graphe 1 pour sa communication avec ND25.

NOTE 1 Les informations de routage de DL dans la charge utile de la DPDU Data sont souvent mises à jour à mesure que cette charge utile passe à travers le sous-réseau de D.

Le Tableau 99 et le Tableau 100 reflètent le contenu des tables de graphes sur ND20 et ND21. Ces tables de graphes correspondent grossièrement à des structures de données au sein de chaque DLE pour la topologie montrée à la Figure 31. Par exemple, une DPDU Data suivant le graphe 2 cherchera l'ID de graphe 2 dans chaque routeur le long du chemin pour savoir quels voisins elle peut utiliser pour le prochain saut.

Tableau 99 – Table de graphes sur ND20

ID de graphe	Adresse du voisin
1	21, 22
2	21, 22

Tableau 100 – Table de graphes sur ND21

ID de graphe	Adresse du voisin
1	23, 24
2	24

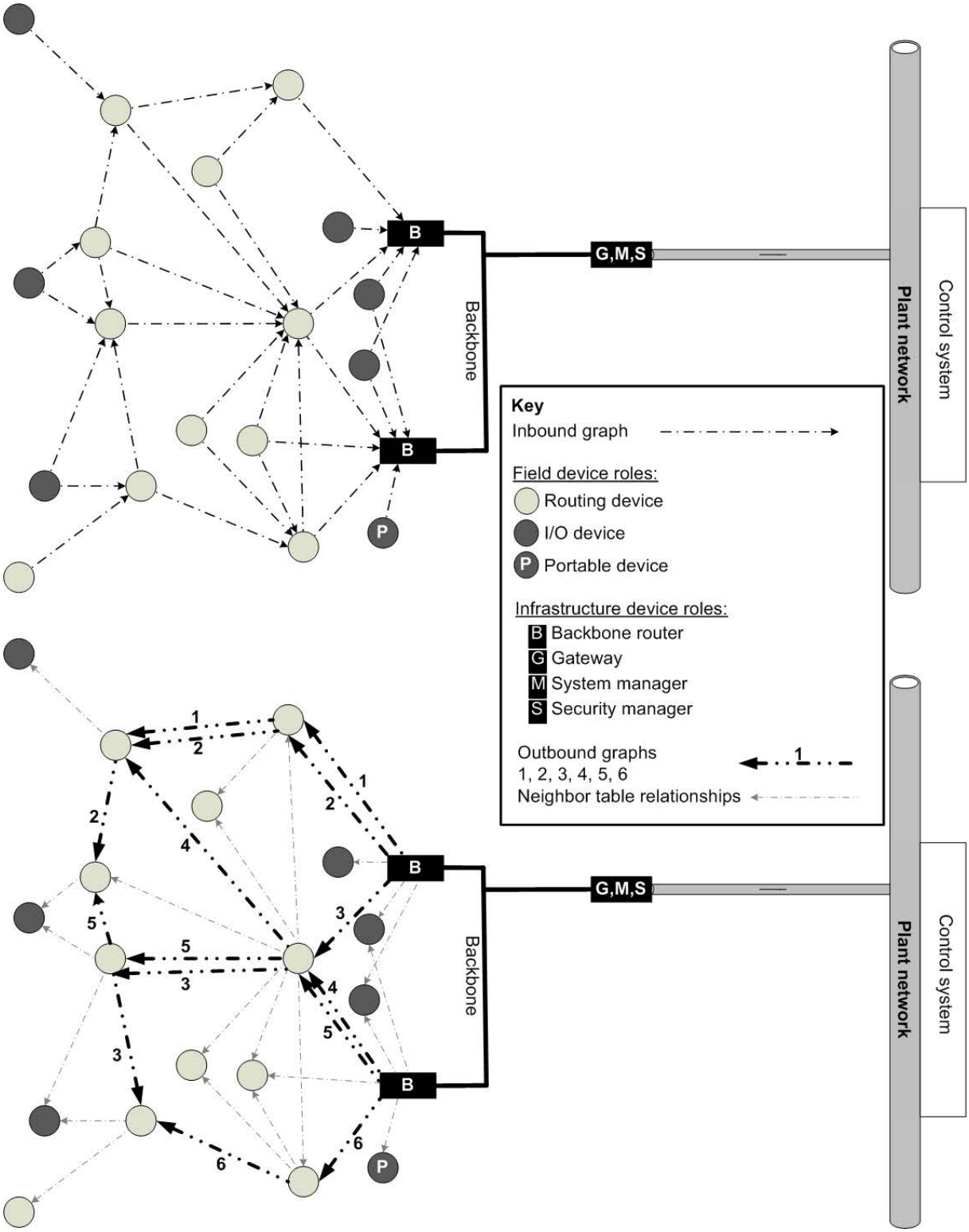
Chaque graphe au sein du sous-réseau D est identifié par un ID de graphe. Une DPDU Data a habituellement son origine au sein du sous-réseau D, à un appareil de terrain ou à une passerelle, un gestionnaire de système ou un routeur dorsal. Pour envoyer un message sur un graphe, la DLE initiatrice inclut un ID de graphe dans le sous-en-tête DROUT de la DPDU Data. La DPDU Data suit les chemins correspondant à l'ID de graphe jusqu'à ce qu'elle atteigne sa destination ou soit rejetée.

Afin d'acheminer des DPDU Data sur un graphe, chaque DLE le long du chemin a besoin de maintenir une table de graphes contenant les entrées qui incluent l'ID de graphe et l'(les) adresse(s) D du(des) prochain(s) voisin(s). Une DLE acheminant une DPDU Data accomplit une consultation basée sur l'ID de graphe et envoie la DPDU Data vers n'importe lequel des voisins applicables. Une fois qu'un voisin acquitte la réception de la DPDU Data, la DLE la libère d'un tampon de transmission de DPDU Data.

Divers chemins de graphe (branches) peuvent être établis en configurant plus d'un voisin associé au même indice de graphe. Une branche peut être configurée avec un voisin préférentiel, indiquant qu'il convient que la DLE tente d'émettre la DPDU Data vers le voisin préférentiel d'abord, même s'il y a une occasion plus précoce d'émettre vers d'autres voisins. Si aucun voisin préférentiel n'est désigné, il convient que la DLE traite équitablement toutes les branches, en émettant la DPDU Data à la première occasion qui se présente. Si la première émission ne donne pas lieu à une DPDU ACK, la DLE est normalement configurée pour utiliser des branches de remplacement pour les répétitions de tentative.

La Figure 55 fournit des exemples de graphes de routage qui sont entrants (allant vers la dorsale) et sortants (s'éloignant de la dorsale). L'organisation de base des graphes de routage entrants et sortants peut être très similaire dans les deux cas, mais en pointant dans des sens opposés, conformément à la Figure 55. Le gestionnaire de système configure les relations de routage entre les DLE dans un sous-réseau D.

La moitié supérieure de la Figure 55 montre un graphe entrant dans une configuration de routage DL exemplaire. Un graphe entrant permet à un ensemble de DLE d'envoyer des DPDU Data vers la dorsale ou le gestionnaire de système. Un seul graphe peut être utilisé pour le routage entrant dans un petit sous-réseau D, comme montré à la moitié supérieure de la Figure 55. Il est possible et souvent souhaitable que le gestionnaire de système définisse plusieurs graphes entrants, en particulier lorsque le nombre des DLE dans le sous-réseau D augmente. Comme montré dans la moitié supérieure de la Figure 55, si des DLE ont plusieurs routeurs voisins, la diversité est souvent inhérente au graphe entrant.



Légende

Anglais	Français
Key	Légende
Inbound graph	Graphe entrant
Field device roles	Rôles d'appareil de terrain
Routing device	Appareil de routage
I/O device	Appareil E/S
Portable device	Appareil portatif

Anglais	Français
Infrastructure device roles	Rôles d'appareil d'infrastructure
B Backbone router	B Routeur dorsal
G Gateway	G Passerelle
M System manager	M Gestionnaire de système
S Security manager	S Gestionnaire de sécurité
Outbound graphs 1,2,3,4,5,6	Graphes sortants
Neighbour table relationships	Relations de la table de voisins
Control system	Système de commande
Plant network	Réseau d'installation
Backbone	Dorsale

Figure 55 – Graphes entrants et sortants

Le graphe entrant illustratif à la Figure 55 peut être problématique pour les NPDU fragmentées, car les fragments arrivant à différents routeurs dorsaux peuvent poser un défi de réassemblage. Les considérations suivantes s'appliquent:

a) Chaque contrat spécifie la taille de NPDU maximale. La plupart des contrats couvrant la communication vers la dorsale, y compris ceux généralement utilisés pour des trafics de plus haute priorité comme la communication P/S des variables de processus et la communication source/puits des alertes (alarmes de processus ou événements de l'appareil par exemple), spécifient une taille d'APDU maximale qui ne déclenche pas la fragmentation des NPDU. La plupart des communications s'adaptent dans une seule et même DPDU Data, qui peut être déterminée pour un flux particulier, en termes de taille maximale de charge utile prise en charge, lorsque le contrat est établi.

b) Les contrats qui permettent des NPDU de taille pouvant impliquer la fragmentation vers la dorsale sont généralement des transferts d'arrière-plan et de faible priorité de grands blocs d'informations, comme le chargement de formes d'ondes capturées. Un problème survient lorsque les fragments d'un NPDU fragmenté sont livrés à des BBR (routeurs dorsaux) différents.

   Ce problème (de livraison des fragments non coordonnée) peut être évité de deux manières différentes:

   1) Le contrat peut spécifier ou être attaché à un chemin qui se termine dans un BBR unique, évitant ainsi la livraison de différents fragments provenant d'un même NPDU à différents appareils.

   2) Le contrat peut spécifier ou être attaché à un chemin qui se termine dans des BBR multiples provenant du même fournisseur, connu par le gestionnaire de système pour avoir un protocole de réassemblage des fragments inter-BBR qui gère ce réassemblage via leurs réseaux dorsaux partagés.

      NOTE 2 Il est courant pour les propriétaires d'usine ou les opérateurs d'acquérir l'équipement des infrastructures remplissant la même fonction chez un seul fournisseur afin que les éventuels problèmes rencontrés avec cette catégorie d'équipement puissent être directement soumis au fournisseur responsable, évitant ainsi au personnel de l'usine d'avoir à déterminer le fournisseur spécifique en cause. Cette élimination de la "désignation de coupables" entre fournisseurs mène généralement à une résolution des problèmes accélérée et au retour à la productivité normale de l'usine plus rapide. Cela rend également plus utile toute caractéristique du fournisseur ou du propriétaire ou tout diagnostic pouvant être fourni par l'équipement des infrastructures.

   Pour les contrats de DLE prenant en charge les tailles de charge utile qui ne s'adaptent pas dans une seule et même DSDU, les graphes sélectionnés orientés vers la dorsale utilisent souvent une diversité de chemin réduite pour assurer qu'un jeu de NPDU fragmentées est livré dans sa totalité au même BBR (alternative a)) ou à un sous-ensemble de BBR multiples connus pour fournir conjointement une capacité partagée de réassemblage de fragments (alternative b)).

La moitié inférieure de la Figure 55 montre un ensemble de graphes sortants dans une configuration de routage exemplaire. Un graphe sortant est habituellement utilisé pour

envoyer des DPDU Data des DLE dorsales vers des DLE de terrain. Comme montré à la moitié inférieure de la Figure 55, plusieurs graphes peuvent être utilisés pour le routage sortant, avec chaque graphe sortant correspondant à un groupe de DLE dans la gamme radio du graphe.

Dans cet exemple, le routage de graphe de DL sortant finit à la dorsale, point auquel la NL prend les responsabilités de routage. La relation entre un sous-réseau D de WISN et un sous-réseau N de dorsale est décrite en 5.5.

Bien que tous les exemples ci-dessus montrent des graphes entrants et sortants, ceux-ci ne sont pas réellement des types de graphes différents, se sont seulement des graphes qui pointent dans des directions opposées. Le routage peer-to-peer est également pris en charge par la norme. Le gestionnaire de système peut configurer un graphe pour suivre n'importe quel chemin où la connectivité existe.

Une DSDU est transmise le long d'un graphe jusqu'à ce que le graphe soit terminé. Si l'adresse de destination de la DPDU Data correspond à l'adresse de la DLE, la DSDU a atteint sa destination et le graphe est terminé. En variante, si le numéro de graphe dans la DPDU Data n'a pas une entrée correspondante dans la table de consultation dlmo.Graph (voir 9.4.3.6), le graphe a atteint son point de fin.

9.1.6.3 Extensions de graphe

La moitié inférieure de la Figure 55 montre que les graphes sortants ne se prolongent pas nécessairement vers toutes les DLE à la périphérie d'un sous-réseau D. Néanmoins, de telles DLE sont implicitement couvertes par les graphes sortants, par le biais d'un mécanisme d'extension de graphe. Une DLE prolonge automatiquement des graphes en vérifiant l'adresse de destination de la DPDU Data comme une entrée de table de voisins, indiquant ainsi que la destination de la DPDU Data se situe à un saut de distance. Si tel est le cas, le routeur traite le voisin comme s'il était énuméré dans le graphe, prolongeant de ce fait le graphe pour la DPDU Data en question. Plus formellement, lorsque l'adresse de destination de la DPDU Data est dans la table de voisins d'une DLE, et la DPDU Data est en cours d'acheminement avec un graphe, la DLE doit traiter ce graphe comme comprenant ce voisin pour le besoin du routage de cette DPDU Data même si le graphe ne se réfère pas explicitement au voisin. Tous les routeurs doivent prendre en charge cette forme élémentaire d'extensions implicites de graphes.

Un champ explicite d'extension de graphe dans la table de voisins fournit un degré supplémentaire de commande. Si un graphe est spécifiquement désigné dans une table de voisins, telle que décrite en 9.4.3.4.2, le voisin est non seulement traité comme étant couvert par le graphe; le voisin reçoit également un traitement préférentiel. Si le voisin est désigné comme étant le dernier saut du graphe, une DPDU Data suivant ce graphe doit être transmise exclusivement à ce voisin. Si le voisin est désigné comme étant une branche préférentielle, il convient que la DLE tente de transmettre une DPDU Data applicable au voisin en question avant les autres voisins.

La prise en charge pour le champ d'extension de graphe explicite dans la table de voisins est une option de construction d'appareil; son statut de prise en charge est rapporté au gestionnaire de système par le biais de dlmo.DeviceCapability lorsque la DLE rejoint le sous-réseau D. Tous les routeurs prennent en charge la capacité d'extension de graphe implicite élémentaire, mais seuls certains routeurs sont censés prendre totalement en charge les indicateurs de dernier saut explicite et de branche préférentielle dans la table de voisins.

9.1.6.4 Routage de source

Le routage de source est une méthode générale de routage prise en charge par la présente norme. Dans le routage de source, la DLE émettrice peu être configurée pour désigner le chemin saut par saut à emprunter par une DPDU Data à travers un sous-réseau D. Une utilisation simple du routage de source correspond au fait de diriger une DPDU Data un saut plus loin à un voisin spécifique, comme pour le rattachement. Lorsqu'une DPDU Data

acheminée par routage de source arrive à une DLE intermédiaire, la DLE intermédiaire examine les informations de chemin contenues dans la DPDU Data pour déterminer le voisin auquel il convient qu'elle transmette la DPDU Data.

Un chemin de source est une liste d'entrées spécifiant le chemin qu'une DPDU Data doit suivre à travers le sous-réseau D. La première entrée dans la liste spécifie le prochain saut, et la liste raccourcit à mesure que la DPDU Data se déplace à travers le sous-réseau D. Les entrées de routage de source peuvent spécifier des graphes ou des adresses D, permettant ainsi à des graphes d'être chaînés. Les numéros de graphe de 12 bits au sein d'un chemin sont codés en binaire comme étant 0x1010 gggg gggg gggg.

L'en-tête de DPDU Data comprime un chemin de source en un seul octet dans le cas commun où un seul chemin de graphe est spécifié et le numéro de graphe est ≤ 255 (codé binaire comme étant 0x1010 0000 gggg gggg). Cela est communément appelé routage par graphe, mais formellement il s'agit d'un chemin de source contenant un seul graphe.

Dans le processus de configuration ou de rattachement, une adresse EUI64Address est utilisée pour adresser une DLE qui n'a pas encore reçu une adresse IPv6Address. Ce cas est codé comme un chemin avec un graphe de zéro et une adresse D de DADDR de zéro (voir 9.3.3.6 et 9.3.3.7), indiquant que l'adresse EUI64Address peut être trouvée dans l'en-tête de MAC (MHR) de la DPDU Data.

Lorsqu'une DPDU Data est reçue par la DLE par sa liaison sans fil, les étapes de traitement suivantes doivent être suivies, dans l'ordre, pour déterminer si le chemin de DL s'est terminé et pour mettre à jour le chemin de source dans l'en-tête de DPDU Data:

- L'adresse D de destination des sous-en-têtes de DADDR est vérifiée pour voir si elle concorde avec l'adresse D de la DLE de réception. S'il y a une concordance, la DSDU a atteint sa destination finale et la DSDU doit être passée à la NLE contiguë comme décrit en 9.2.4. (L'adresse D de destination de DADDR est codée comme étant zéro, dans le cas où l'adresse D de destination serait dupliquée dans le MHR. Voir 9.3.3.7. Dans ce cas, la concordance est indirecte, basée sur l'adresse D de la destination de MHR.)
- La première entrée dans le chemin de source est supprimée s'il y a lieu, raccourcissant donc le chemin de source en décalant la deuxième entrée et les suivantes (le cas échéant) vers les positions antérieures (décalage à gauche). La première entrée doit être supprimée, à moins qu'elle ne soit un numéro de graphe pour un graphe qui n'a pas atteint son point de fin.
- Si le chemin n'a aucune entrée restante, le chemin est terminé et la DSDU doit être passée à la NLE contiguë comme décrit en 9.2.4.

Si la DSDU n'est pas passée à la NLE contiguë, la DPDU Data doit être rejetée si la limite de transmission (dans le sous-en-tête DROUT) est zéro. Si la limite de transmission est positive, elle est décrémentée et la DPDU Data est placée sur la file d'attente de messages de transmission de la DLE.

Lorsqu'une DSDU est prévue pour être acheminée par la dorsale, il convient que le chemin de DL se termine au routeur dorsal. Si un chemin ne se termine pas dans un routeur dorsal contigu, elle est transmise par la DLE et n'est jamais traitée par la NLE contiguë du routeur dorsal, permettant donc à la messagerie peer-to-peer de se produire au sein du sous-réseau D sous les auspices d'un routeur dorsal.

Ces méthodes de routage sont des exemples de différentes manières de configurer la capacité de routage DL qui réside dans tous les routeurs de terrain conformes à la présente norme. Le gestionnaire de système doit configurer tous les graphes au sein d'un sous-réseau D. La capacité de configurer le routage de plusieurs manières différentes telles que le routage par graphe et/ou le routage de source permet l'interconnectabilité d'appareil.

9.1.6.5 Sélection de chemin

Le chemin à travers le sous-réseau D pour une DPDU Data est choisi lorsque le message pénètre dans la DL (voir 9.2.2). Le chemin est stocké dans le sous-en-tête DROUT pour être utilisé par d'autres DLE qui achemineront la DPDU Data. La sélection initiale d'un chemin est basée sur des règles de décision dans la DLE initiatrice. La liste suivante montre les critères de sélection de chemin, dans l'ordre, par lesquels le chemin doit être sélectionné en se basant sur la première condition qui s'applique:

- La DPDU Data a une adresse de destination EUI64Address. Une adresse de destination EUI64Address est utilisée seulement au cours du processus de rattachement, lorsqu'un routeur envoie une réponse à un voisin immédiat qui n'a pas encore reçu une adresse DL16Address issue du gestionnaire de système. Dans ce cas, l'adresse EUI64Address issue de l'IEEE MAC est utilisée, avec un ID de graphe (Graph ID) de 0 dans le chemin D.
- Le ContractID est associé à un chemin D particulier. Il peut être utilisé lorsqu'un graphe ou chemin D de source particulier est prévu pour fournir un niveau défini de service.
- L'adresse de destination DL16Address au sein du sous-réseau D est associée à un chemin particulier.
- L'adresse de destination DL16Address au sein du sous-réseau D est un voisin immédiat, comme au cours du processus de rattachement.
- Autrement, utiliser le chemin par défaut. Normalement, la valeur par défaut dirigera le message vers le routeur dorsal le plus proche, ou vers le gestionnaire de système s'il n'y a aucun routeur dorsal.

Un seul chemin peut être désigné comme étant la valeur par défaut par le gestionnaire de système, en désignant un chemin D particulier comme étant la valeur par défaut. Le chemin D par défaut est habituellement configuré pour acheminer des messages vers le gestionnaire de système, ou vers un routeur dorsal s'il n'y a aucun gestionnaire de système sur le sous-réseau D. Un chemin D par défaut peut raisonnablement être configuré conjointement avec l'établissement du contrat d'une DLE avec le gestionnaire de système. Des chemins complémentaires peuvent être configurés selon les besoins, comme pour fournir une qualité de service accrue ou pour acheminer des messages vers une DLE homologue sur le sous-réseau D.

9.1.7 Saut de voie discrétisé, saut de voie lent et intervalles de temps

9.1.7.1 Généralités

Trois variantes opérationnelles générales sont prises en charge par la DL:

– saut de voie discrétisé;

– saut de voie lent; et

– combinaisons hybrides de saut de voie discrétisé et de saut de voie lent.

Ces trois variantes opérationnelles sont différentes manières pour un gestionnaire de système de configurer une capacité de saut de voie discrétisé qui est prise en charge par chaque DLE dans un sous-réseau D. Des programmes de saut de voie sont configurés par le gestionnaire de système par l'intermédiaire des DPDU d'annonce et de l'attribut dlmo.Superframe.

Le saut de voie discrétisé et le saut de voie lent fournissent différentes manières de configurer une série d'intervalles de temps. Le gestionnaire de système détermine le mode de fonctionnement et assigne l'utilisation des supertrames, qui sont des collections cycliques d'intervalles de temps. (Voir 9.1.8 pour un débat plus poussé relatif aux supertrames.) Cela fournit la flexibilité et l'interopérabilité des fonctions de communication pertinentes (l'interconnectabilité par exemple) sans exiger une complexité excessive au sein des appareils.

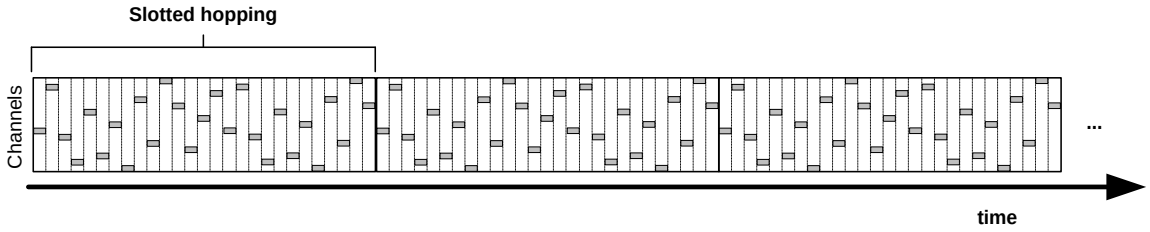
De la perspective d'un appareil de terrain, la DLE peut être visualisée comme un piano avec plusieurs touches. Chaque touche correspond à une transaction D, qui spécifie une voie et un intervalle de temps spécifiques. Une touche est utilisée pour envoyer une DPDU Data en attente issue de la file d'attente sortante, une autre touche est utilisée pour être à l'écoute pour détecter une DPDU Data entrante, et ainsi de suite. Le gestionnaire de système fournit une bande perforée pour piano pour que la DLE joue sans cesse. Le style de jeu peut être saut de voie lent, saut de voie discrétisé, ou un hybride des deux. La DLE ne fait pas de différence; elle joue simplement mécaniquement chaque touche à un moment spécifié sur une voie spécifiée, en se basant sur les instructions contenues sur la bande perforée pour piano.

Les détails de note au sein d'un intervalle de temps sont configurables, en utilisant des modèles d'intervalles de temps fournis par le gestionnaire de système. Il y a une série contrainte d'opérations qui peuvent être accomplies au sein d'un intervalle de temps – émission, écoute, attente, temporisation et acquittement – mais ces blocs modules de base simples peuvent être assemblés en des temps différents. Ces définitions sont flexibles, sous le contrôle du gestionnaire de système.

La durée des intervalles de temps dans un sous-réseau D est mise à une valeur spécifique par le gestionnaire de système lorsqu'une DLE rejoint le sous-réseau D. Une durée d'intervalle de temps de 10 ms à 12 ms est censée être typique. La durée d'intervalle de temps est configurable pour permettre:

- la coexistence optimisée avec d'autres systèmes, tels que d'autres sous-réseaux D conformes à la présente norme, à l'IEC 62591, et à l'IEC 62601;
- de plus longs intervalles de temps pour prendre en charge des temps prolongés d'attente de message;
- de plus courts intervalles de temps pour tirer pleinement profit de mises en œuvre optimisées;
- de plus longs intervalles de temps pour prendre en charge des DPDU ACK/NAK DPDU en série issues de plusieurs appareils (par exemple: duodiffusion, N-diffusion);
- de plus longs intervalles de temps pour prendre en charge des accès CSMA/CA de longue durée au début d'un intervalle de temps (par exemple: pour l'accès hiérarchisé par ordre de priorité à des intervalles de temps partagés);
- de plus longs intervalles de temps pour prendre en charge des périodes de saut lent de durée prolongée;
- des intervalles de temps pour la synchronisation avec d'autres sous-réseaux non conformes et faciliter l'inter-routage.

La Figure 56 montre le fonctionnement du saut de voie discrétisé.



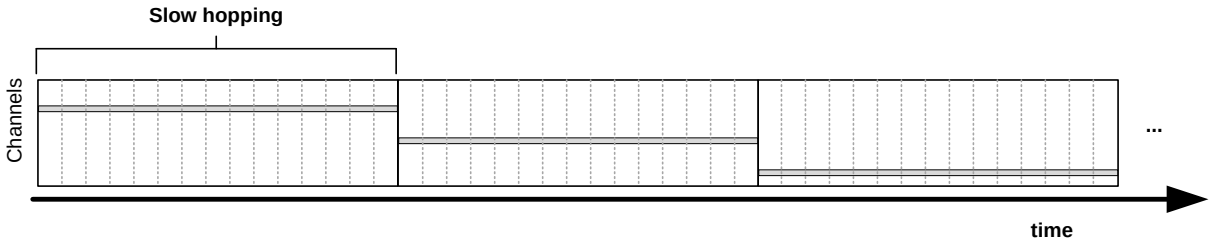
Légende

Anglais	Français
Slotted hopping	Saut discrétisé
Channels	Voies
Time	Temps

Figure 56 – Saut de voie discrétisé

Dans le saut de voie discrétisé, des intervalles de temps de saut de voie d'égale durée sont utilisés. Chaque intervalle de temps utilise une voie radio différente dans un modèle de saut de voie. Dans le saut de voie discrétisé, chaque intervalle de temps est prévu pour prendre en charge une seule transaction D constituée d'une DPDU Data et de son(ses) acquittement(s) de DPDU ACK/NAK.

La Figure 57 montre le fonctionnement du saut de voie lent.



Légende

Anglais	Français
Slow hopping	Saut lent
Channels	Voies
Time	Temps

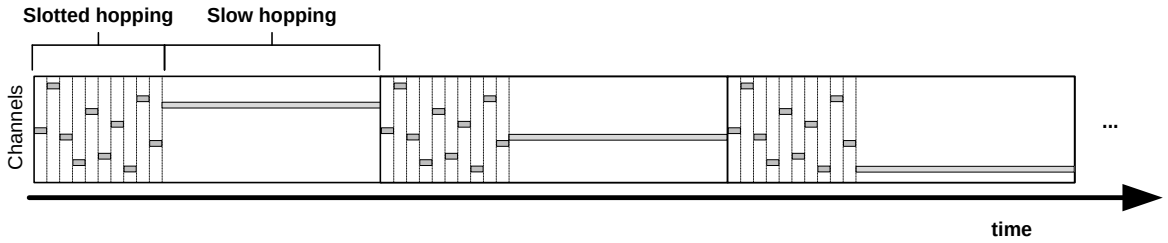
Figure 57 – Saut de voie lent

Dans le saut de voie lent, un ensemble d'intervalles de temps contigus est groupé sur une seule voie radio réelle. Chaque ensemble d'intervalles de temps est traité comme une seule période de saut de voie lent; cependant, conformément à la Figure 57, les intervalles de temps sous-tendent encore le saut de voie lent. Les périodes de saut de voie lent sont configurables, habituellement sur une échelle d'environ 100 ms par saut à 400 ms par saut. De plus longues périodes de saut de voie lent, potentiellement d'une durée de plusieurs secondes, peuvent être utilisées pour prendre en charge des DLE avec un chronométrage imprécis et/ou des DLE qui ont temporairement perdu le contact avec le sous-réseau D. Pour permettre l'interopérabilité de DLE, toutes les DLE conformes à la présente norme doivent prendre en charge la configuration de la durée d'intervalle de temps, telle que désignée par le gestionnaire de système.

Dans certains domaines de réglementation, le saut de voie lent d'une radio dans la bande de 2,4 GHz selon l'IEEE 802.15.4 n'est pas autorisé à excéder 400 ms par saut. Cependant, dans d'autres domaines de réglementation, il n'y a aucune contrainte de ce type. La période de saut de voie lent est fixée par le gestionnaire de système, pas par la présente norme, mais son utilisation est contrainte dans chaque DLE par dlmo.CountryCode (9.1.15.6). Ainsi, cette contrainte de réglementation est explicitement appliquée où elle existe, comme d'autres contraintes de réglementation.

La structure, la durée, et l'assignation des intervalles de temps dans les périodes de saut de voie lent sont configurées par le gestionnaire de système, qui détermine le mode de fonctionnement et assigne l'utilisation des intervalles de temps. Cela fournit la flexibilité sans exiger une complexité excessive au sein des DLE, facilitant l'interfonctionnabilité des DLE.

La Figure 58 montre un mode de fonctionnement hybride.



Légende

Anglais	Français
Slotted hopping	Saut discrétisé
Slow hopping	Saut lent
Channels	Voies
Time	Temps

Figure 58 – Fonctionnement hybride

Le fonctionnement hybride utilise de périodes de saut de voie discrétisé et de saut de voie lent dans une combinaison configurée. Par exemple, dans la Figure 58, un certain nombre d'intervalles de temps utilisant le saut de voie discrétisé est suivi d'une période de saut de voie lent.

9.1.7.2 Saut de voie

9.1.7.2.1 Généralités

Des communications de DL sont destinées à être distribuées à travers plusieurs voies radio. Le système utilise des modèles définis de saut de voie qui fournissent une séquence spécifique de voies pour la communication parmi des ensembles d'appareils. Le saut de voie commence à un décalage désigné dans le modèle de saut de voie, en continuant à travers le modèle séquentiellement jusqu'à la fin, puis en répétant le modèle indéfiniment.

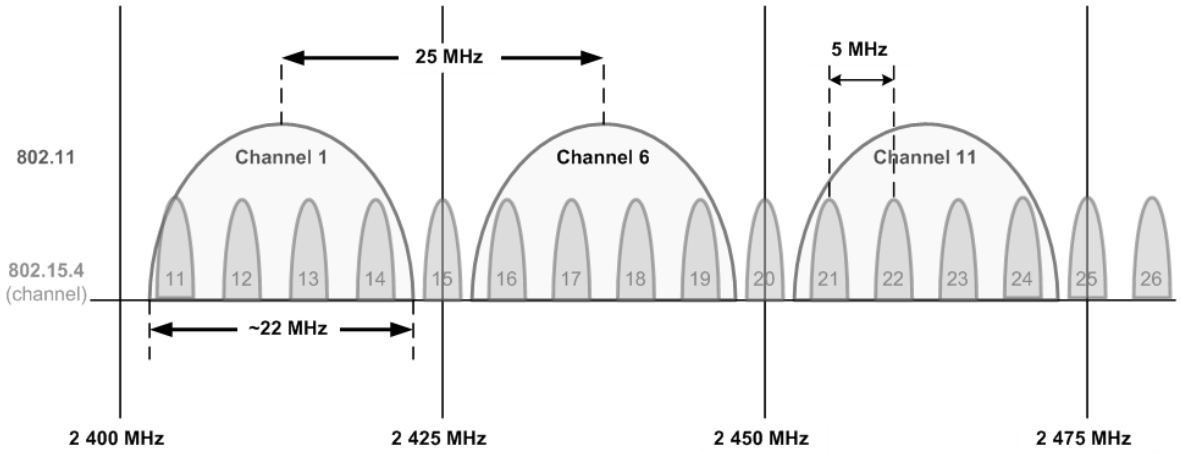
Les voies 11..26 DSSS selon l'IEEE 802.15.4 sont mappées à des numéros nominaux de voie 0..15 dans la présente norme. Cette vue d'ensemble leur fait référence par leur nomenclature de l'IEEE 802.15.4, comme voies 11..26.

La présente norme est basée sur les appareils qui utilisent une seule voie à la fois. Les radios multiples qui sont en copaquetage sont capables d'opérer plus d'instances de la PhLE et de la DLE simultanément sur différentes voies. Les détails d'une telle opération sur plusieurs voies ne sont pas spécifiés par la présente norme, mais une telle opération est intentionnellement prise en charge dans le nonce D.

9.1.7.2.2 Considérations de spectre radio

Pour la radiocommunication, la présente norme utilise les voies DSSS dans la bande de 2,4 GHz selon l'IEEE 802.15.4. La couche physique de l'IEEE 802.15.4 (DSSS à 2,4 GHz) inclut seize voies, numérotées de 11 à 26.

La Figure 59 montre les seize voies DSSS selon l'IEEE 802.15.4 avec trois voies selon l'IEEE 802.11 communément utilisées qui se chevauchent.



Légende

Anglais	Français
802.15.4 (channel)	802.15.4 (voie)
Channel 1	Voie 1
Channel 6	Voie 6
Channel 11	Voie 11

NOTE Les numéros de voie montrés sont ceux de l'IEEE 802.11 et de l'IEEE 802.15.4, au lieu de ceux de la présente norme.

Figure 59 – Utilisation du spectre radio

A la Figure 59, les voies étroites 11..26 sont des voies DSSS dans la bande de 2,4 GHz selon l'IEEE 802.15.4; ces voies sont essentiellement non superposées. Sont également montrées les voies plus larges 1, 6 et 11 de l'IEEE 802.11; ces trois voies sont des choix communs pour la communication selon l'IEEE 802.11 et chacune chevauche un certain nombre de voies DSSS dans la bande de 2,4 GHz selon l'IEEE 802.15.4.

Comme la Figure 59 le montre, les voies 15, 20 et 25 DSSS dans la bande de 2,4 GHz selon l'IEEE 802.15.4: ne chevauchent essentiellement aucune des trois voies communes de l'IEEE 802.11. Par conséquent, les voies 15, 20 et 25 DSSS dans la bande de 2,4 GHz selon l'IEEE 802.15.4 peuvent raisonnablement être désignées comme étant des voies à saut de voie lent. Ces voies à saut de voie lent peuvent être configurées pour des besoins tels que la découverte de voisins. Par exemple, des informations relatives au sous-réseau D peuvent être annoncées par des routeurs de terrain sur des voies prédésignées à saut de voie lent, afin que toute DLE cherchant à rejoindre le sous-réseau D puisse limiter ses balayages actifs et passifs à ces voies lors de la découverte de routeurs de terrain limitrophes.

Cette illustration démontre comment des techniques de gestion de spectre prises en charge par la norme peuvent être utilisées pour rendre compte d'un scénario communément rencontré. Des configurations alternatives de WiFi, d'autres utilisateurs du spectre radio, et des restrictions de réglementation locales peuvent justifier des configurations alternatives dans les installations réelles.

La plupart des communications de DL sont destinées à être distribuées à travers toutes les voies (11..26) DSSS de l'IEEE 802.15.4 qui sont disponibles. La présente norme définit un certain nombre de modèles prédéfinis de saut de voie, chacun fournissant une séquence spécifique de voies à utiliser. D'autres modèles de saut de voie sont configurables par le gestionnaire de système. Les modèles de saut de voie qui sont prédéfinis dans la présente norme sont sélectionnés pour avoir certaines propriétés visant à réduire au maximum l'occurrence de collisions répétitives et non gérées avec des appareils sans fil contigus, en particulier des appareils IEEE 802.11.

9.1.7.2.3 Voie 26 et autres voies bloquées

La prise en charge de la voie 26 DSSS dans la bande de 2,4 GHz selon l'IEEE 802.15.4 est facultative dans la présente norme, car certaines mises en œuvre peuvent rencontrer des restrictions réglementaires à la frontière supérieure de la bande. En outre, une DLE peut être empêchée d'utiliser d'autres voies en raison de restrictions réglementaires, mais pas pour d'autres raisons. Après la restriction ou une activation forcée pour le domaine de réglementation spécifié par dlmo.CountryCode (9.1.15.6), une liste de voies que la DLE prend en charge est rapportée au gestionnaire de système au cours du processus du rattachement de la DLE au sous-réseau D. Cela est accompli par le truchement de l'attribut dlmo.DeviceCapability.ChannelMap. Une DLE peut être configurée avec des liaisons qui utilisent de telles voies non prises en charge; dans ce cas, la DLE doit traiter ces liaisons comme étant non sélectionnables.

Sachant que la prise en charge de la voie 26 DSSS dans la bande de 2,4 GHz selon l'IEEE 802.15.4 est facultative dans la norme, un gestionnaire de système peut raisonnablement limiter le fonctionnement de sous-réseau D aux voies 11..25 DSSS dans la bande de 2,4 GHz selon l'IEEE 802.15.4. Les modèles de saut de voie prédéfinis par la présente norme incluent la voie 26 DSSS dans la bande de 2,4 GHz selon l'IEEE 802.15.4, mais ces séquences prédéfinies de saut de voie sont conçues de sorte qu'elles puissent être raccourcies en excluant de la carte de voies dans chaque supertrame la voie 26 DSSS dans la bande de 2,4 GHz selon de l'IEEE 802.15.4.

9.1.7.2.4 Gestion de spectre et utilisation sélective de voie

Plusieurs méthodes sont disponibles pour limiter l'utilisation des voies radio occupées ou indésirables, y compris l'évaluation de voie libre (CCA), la gestion de spectre, et l'utilisation sélective de voie.

Des intervalles de temps sont normalement configurés pour vérifier l'existence d'une voie libre avant émission, en utilisant le mécanisme de la CCA défini dans l'IEEE 802.15.4. La CCA amène une DLE qui est sur le point d'initier une émission à abandonner un intervalle de temps si l'utilisation de la voie par une autre DLE est détectée avant l'émission. Voir 4.6.11, 9.1.9.4.3 et 9.1.9.4.8.

La gestion de spectre est une forme d'utilisation sélective de voie. La gestion de spectre limite la configuration de DL à un sous-ensemble de voies. Le fait de limiter le saut de voie lent aux voies 15, 20 et 25 de DSSS dans la bande de 2,4 GHz selon l'IEEE 802.15.4 est un exemple de gestion de spectre. Un autre exemple est lorsqu'un gestionnaire de système bloque (place en liste noire) certaines voies radio qui ne fonctionnent pas bien ou sont interdites par la réglementation ou la politique locale, ou place en liste blanche des voies qui sont mandatées par la réglementation ou la politique locale. La gestion de spectre est traitée par le gestionnaire de système, par la manière dont il configure une DLE et la PhLE associée. Voir 9.1.8.4.7.

En plus, une DLE peut en toute autonomie traiter des liaisons d'émission sur des voies problématiques comme étant "idle", réduisant ainsi les interférences inutiles et l'énergie gaspillée sur des voies ayant des antécédents de connectivité médiocre. Il convient qu'une DLE sautant des liaisons de cette manière soumette périodiquement les liaisons à un essai afin de vérifier qu'elles restent problématiques. Une telle utilisation sélective de voie peut être désactivée par le gestionnaire de système sur une base liaison par liaison, par le biais de l'attribut dlmo.Link[ ].Type.SelectiveAllowed. Voir 9.4.3.7.2 et Tableau 182.

9.1.7.2.5 Modèles répétitifs de saut de voie

La présente norme prend en charge les modèles répétitifs prédéfinis de saut de voie de DSSS dans la bande de 2,4 GHz selon l'IEEE 802.15.4:2011, qui doivent être pris en charge dans chaque DLE:

- pattern1 (modèle1): 19, 12, 20, 24, 16, 23, 18, 25, 14, 21, 11, 15, 22, 17, 13 (, 26);

- pattern 2: pattern1 dans l'ordre inverse;
- pattern3 (modèle3): 15, 20, 25 (destiné aux voies à saut de voie lent);
- pattern4 (modèle4): 25, 20, 15 (pattern3 dans l'ordre inverse);
- pattern5 (modèle5): 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21. 22, 23, 24, 25 (, 26).

NOTE 1 La voie 26 de DSSS dans la bande de 2,4 GHz selon l'IEEE 802.15.4 est montrée entre parenthèses, car elle est prise en charge par la présente norme, mais elle n'est pas nécessairement utilisée en raison de contraintes réglementaires communément rencontrées à la frontière de la bande. De la Figure 60 à la Figure 66 et de la Figure 70 à la Figure 74, les figures incluent pour la plupart la voie 26 bien que son utilisation soit généralement masquée par la supertrame qui utilise la séquence de sauts.

NOTE 2 Dans la présente norme, les voies sont numérotées 0..15, conformément à la description en 9.4.3.2. Cependant, pour des besoins tutoriaux et pour faciliter la comparaison avec l'IEEE 802.11 (WiFi) et d'autres utilisations de la même bande de fréquences, les modèles de saut de voie sont exprimés par leurs numéros de voies DSSS selon l'IEEE 802.15.4.

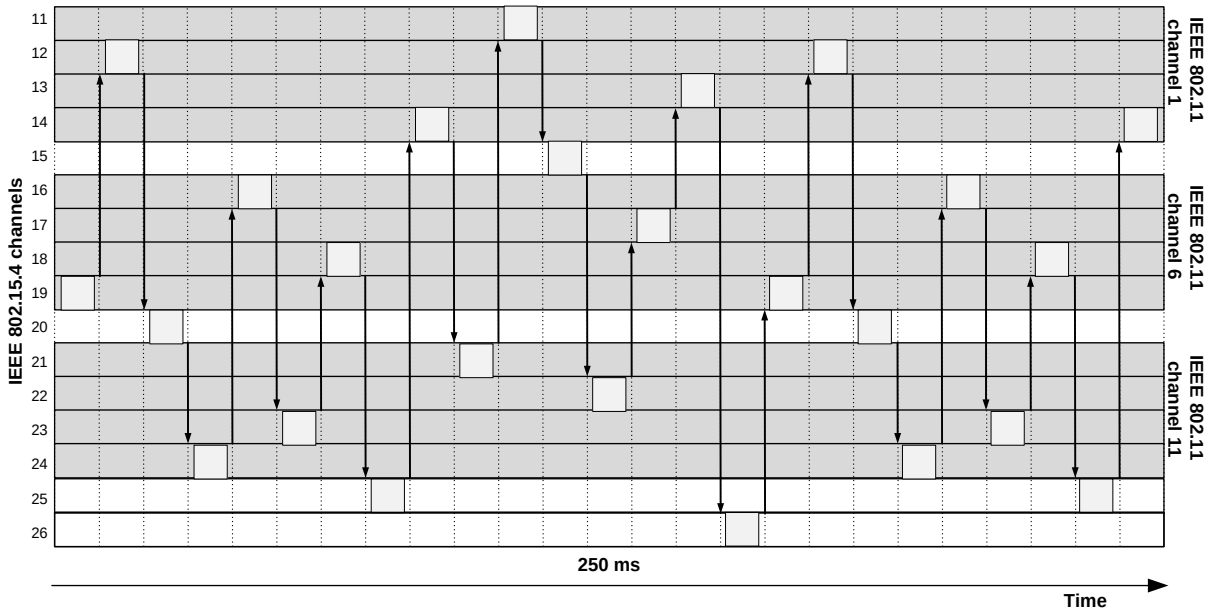
NOTE 3 Pattern5, qui est basé sur la IEC 62591, vise à faciliter la coexistence avec la norme IEC en question.

Le gestionnaire de système peut configurer une DLE pour utiliser l'un quelconque de ces modèles de saut de voie pour le saut de voie discrétisé, le saut de voie lent, ou le saut de voie hybride.

Une voie ou un ensemble quelconque de voies dans un modèle de saut de voie peut être déactivé(e) (masqué(e)) par la configuration de la supertrame qui utilise le modèle de saut de voie, avec l'effet de raccourcir le modèle de saut de voie pour la gestion de spectre.

Les modèles prédéfinis de saut de voie de la présente norme sont conçus pour avoir certaines propriétés, indépendamment du fait que la voie 26 soit incluse dans le modèle ou pas. Spécifiquement, des voies successives dans la plupart des modèles prédéfinis de saut de voie sont séparées d'au moins 15 MHz – trois voies DSSS dans la bande de 2,4 GHz selon l'IEEE 802.15.4. Cette propriété atténue les effets des interférences et de l'évanouissement par trajets multiples dans les environnements industriels.

Comme montré dans l'exemple à la Figure 60, le modèle prédéfini pattern1 de saut de voie est configuré de façon que des sauts consécutifs ne chevauchent pas la même voie selon l'IEEE 802.11.



Légende

Anglais	Français
IEEE 802.15.4 channels	Voies de l'IEEE 802.15.4

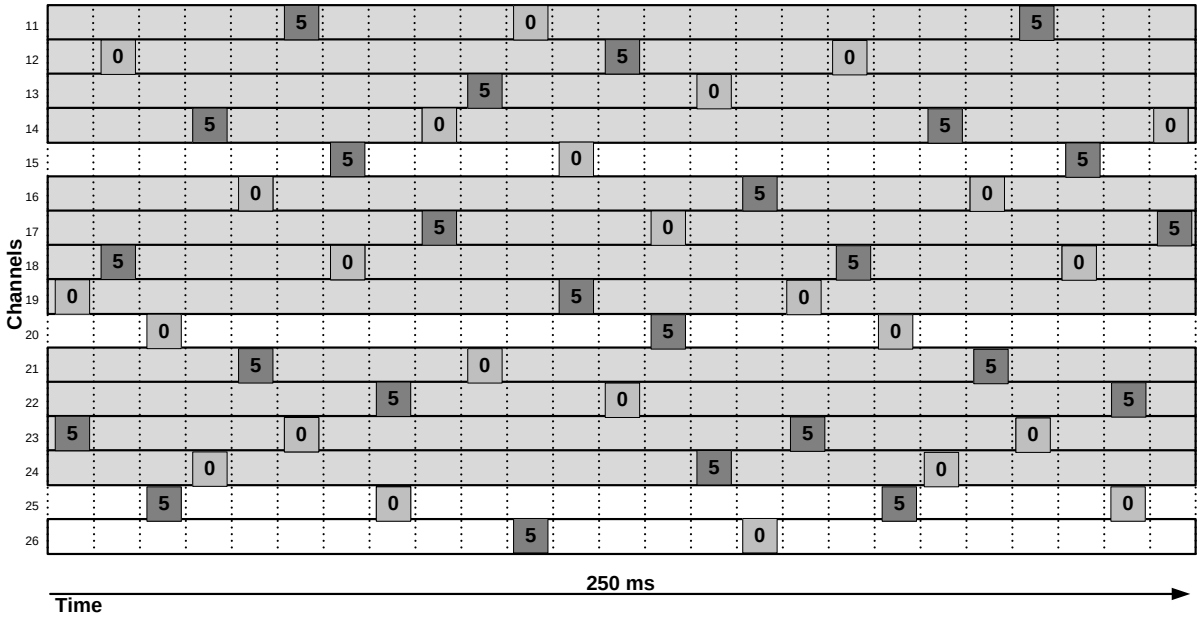
Anglais	Français
IEEE 802.11 channel 1	Voie 1 de l'IEEE 802.11
IEEE 802.11 channel 6	Voie 6 de l'IEEE 802.11
IEEE 802.11 channel 11	Voie 11 de l'IEEE 802.11
Time	Temps

Figure 60 – Modèle prédéfini de saut de voie 1

Au moins trois voies séparent chaque saut consécutif dans le modèle pattern1, conduisant à des décalages de fréquence d'au moins 20 MHz. Lorsque des répétitions de tentative se produisent en des sauts consécutifs, elles ne rencontreront pas ou ne causeront pas d'interférences dans la même voie IEEE 802.11.

Pour de groupes de DLE différents, il est souhaitable que les DLE sautent sur des modèles sans interférence.

Chaque modèle de saut de voie est combiné à un décalage de modèle de saut. Si le décalage de modèle de saut est zéro, le modèle spécifié de saut de voie de ligne de base est utilisé. Si le décalage de modèle de saut est 5, un décalage de 5 est utilisé pour indexer le modèle de saut de voie de ligne de base. La Figure 61 montre comment deux groupes de DLE avec des décalages de modèle de saut différents dans le modèle de saut de voie pattern1 peuvent être utilisés ensemble sans se disputer en même temps la même voie radio.



Légende

Anglais	Français
Channels	Voies
Time	Temps

NOTE Les numéros de voie montrés sont ceux de l'IEEE 802.15.4, au lieu de ceux de la présente norme.

Figure 61 – Deux groupes de DLE avec différents décalages de modèle de saut de voie

Le décalage de saut de voie d'une supertrame est déterminé indirectement à partir de l'attribut dlmo.Superframe[ ].ChBirth, appelé simplement ChBirth, qui donne la référence de début du modèle de saut de voie au temps TAI zéro. Cela fournit une séquence de saut de voie de ligne de base. Les décalages issus d'une séquence de saut de voie de ligne de base, spécifiés dans l'attribut de dlmo.Link[ ].ChOffset, appelé simplement ChOffset, sont tels que décrits ici et montrés à la Figure 61. Alors que le même résultat est réalisable en utilisant

ChBirth ou ChOffset, il convient de noter que les deux attributs sont essentiellement inverses. Les détails des calculs pour le décalage de modèle de saut de voie sont donnés en 9.4.3.5.3.

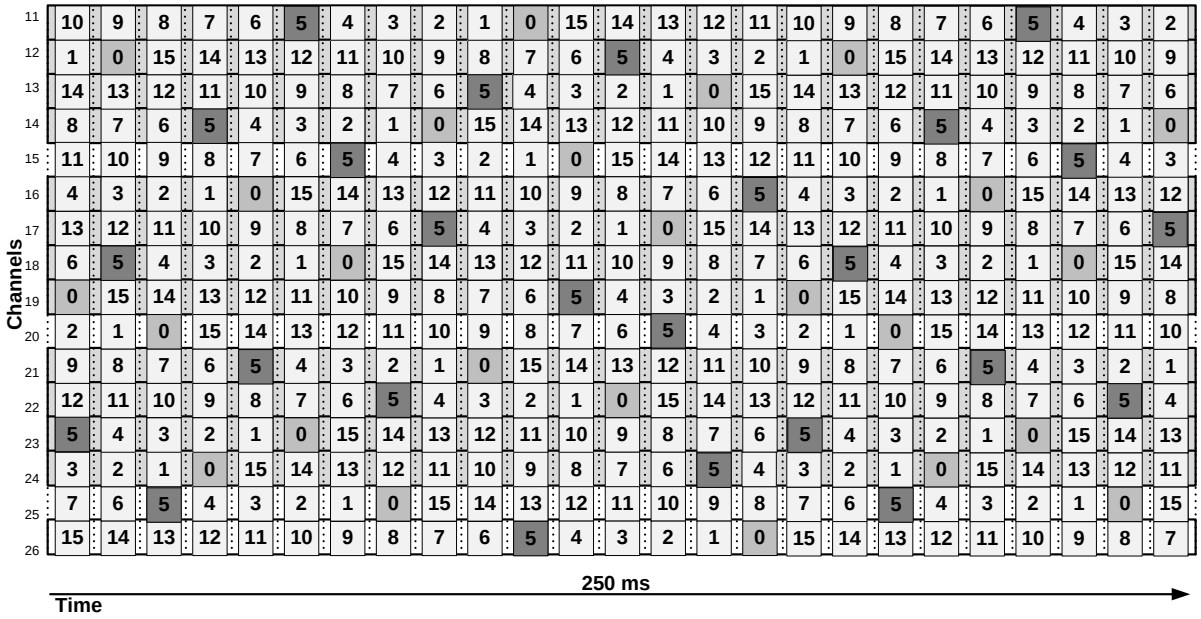
A la Figure 61, les cases numérotées 0 représentent un groupe de DLE utilisant le modèle prédéfini de saut de voie, pattern1; son modèle répétitif de saut de voie (avec la nomenclature de l'IEEE 802.15.4) est:

19, 12, 20, 24, 16, 23, 18, 25, 14, 21, 11, 15, 22, 17, 13, 26

Les cases numérotées 5 à la Figure 61 représentent un autre groupe de DLE utilisant le modèle de saut de voie pattern1 avec un décalage de modèle de saut de voie de 5. Un décalage de modèle de saut de voie de 5 a l'effet de tourner essentiellement la séquence de saut de voie vers la gauche de 5, conduisant à une séquence répétitive de saut de voie (avec la nomenclature de l'EEE 802.15.4) de:

23, 18, 25, 14, 21, 11, 15, 22, 17, 13, 26, 19, 12, 20, 24, 16

La Figure 62 étend le principe, montrant comment des décalages différents de modèle de saut de voie peuvent être utilisés pour un plus grand nombre de DLE.



Légende

Anglais	Français
Channels	Voies

NOTE Les numéros de voie montrés sont ceux de l'IEEE 802.15.4, au lieu de ceux de la présente norme.

Figure 62 – Pattern1 entrelacé de saut de voie avec seize différents décalages de modèle de saut de voie

A la Figure 62, seize voies sont disponibles. Par conséquent, le nombre maximal des DLE qui peuvent utiliser le pattern1 de saut de voie est 16, chacune avec un différent décalage de modèle de saut de voie de 0..15.

Comme montré à la Figure 62, un modèle donné de saut de voie peut être utilisé simultanément en assignant des décalages différents de modèle de saut de voie à diverses DLE, à diverses supertrames (voir 9.1.8) ou à divers groupes de DLE. A titre de simple exemple, chacun de deux groupes de DLE peut utiliser le même modèle de saut de voie avec différents décalages de modèle de saut de voie et, ainsi, les deux groupes peuvent partager le même spectre radio sans interférence mutuelle. Les groupes peuvent être situés dans le même sous-réseau D ou dans des sous-réseaux D différents; tant qu'ils partagent de façon

exacte une durée cohérente d'intervalle de temps et un sens synchronisé du temps, leurs modèles de saut de voie peuvent être entrelacés comme montré à la Figure 62.

Les cinq modèles prédéfinis de saut de voies doivent exister dans chaque DLE conforme à la présente norme. Cela permet à des routeurs d'annoncer avec concision le modèle de saut de voie qui est utilisé et le décalage de modèle de saut de voie courant, lorsqu'il est l'un de ces modèles. La présente norme prend également en charge les modèle personnalisés de saut de voie dans chaque DLE, en plus des cinq modèles prédéfinis, et le gestionnaire de système peut ainsi configurer des modèles complémentaires de saut de voie utilisables par les DLE déjà rattachées.

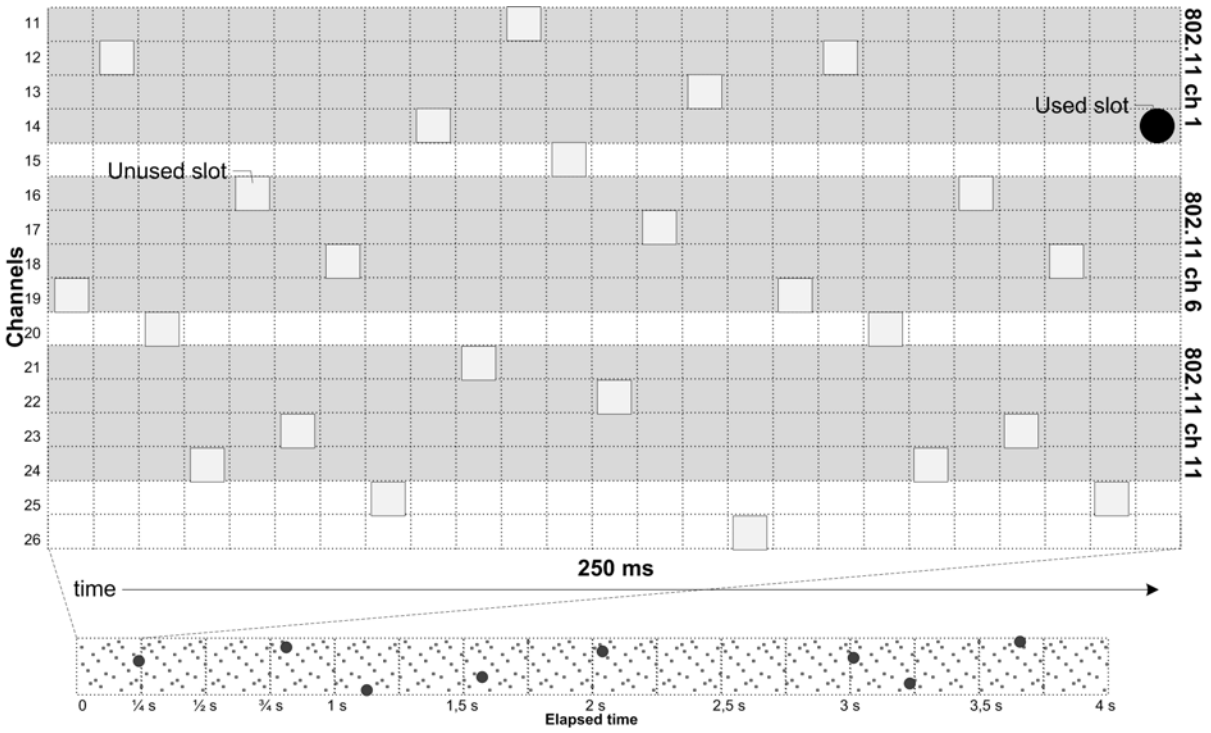
Chaque modèle de saut de voie prédéfini a la même taille que le nombre de voies utilisées. Ainsi, par exemple, le pattern1 de saut de voie utilise seize voies et a donc une longueur de seize sauts. Cette propriété permet au modèle de saut de voie d'être entrelacé à différents décalages, comme montré à la Figure 62.

NOTE 4 Sachant que certaines DLE pourraient ne pas prendre en charge la voie 26, qui est facultative, des systèmes limitent souvent le fonctionnement à quinze voies (11..25) avec essentiellement le même résultat.

9.1.7.2.6 Utilisation d'intervalle de temps et de voie

Un système doit utiliser le saut de voie discrétisé, le saut de voie lent ou une combinaison hybride des deux.

La Figure 63 montre l'utilisation du saut de voie discrétisé. Chaque intervalle de temps est utilisé avec la prochaine voie successive dans le modèle de saut de voie.



Légende

Anglais	Français
Channels	Voies
Used slot	Intervalle utilisé
Unused slot	Intervalle non utilisé
Time	Temps
Elapsed time	Temps écoulé

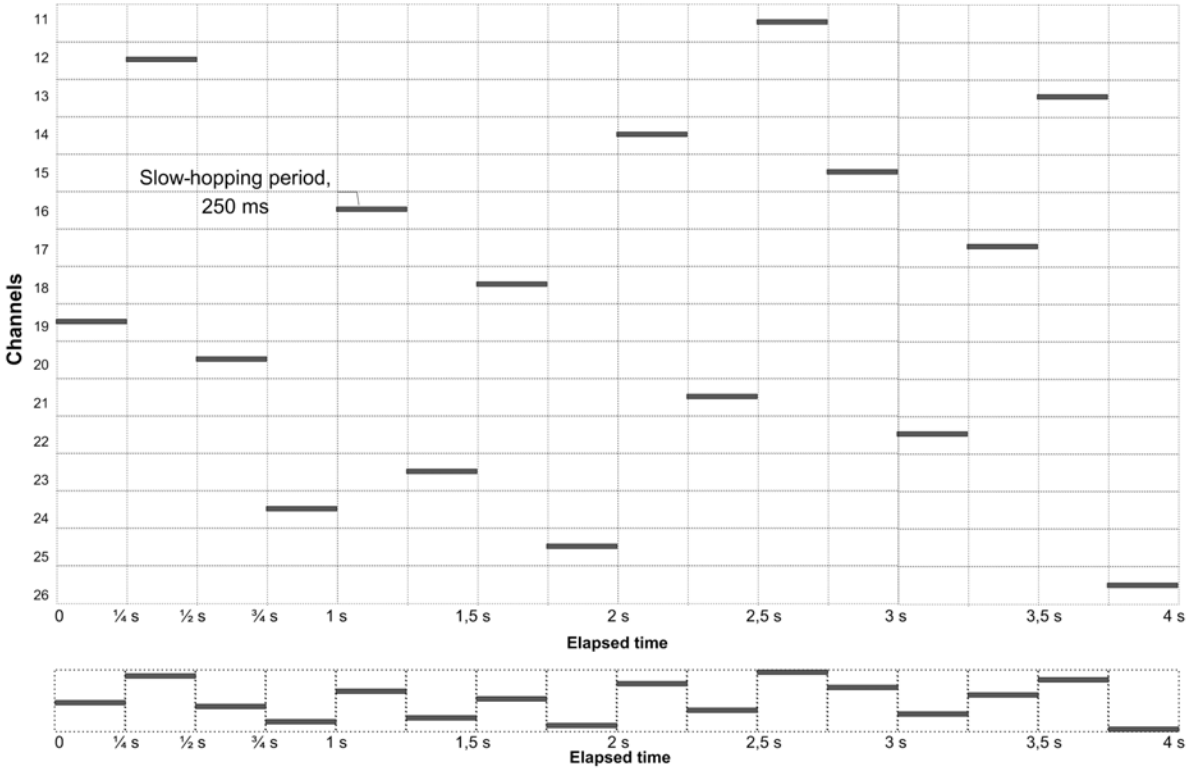
Anglais	Français
802.11 ch 1	802.11 voie 1
802.11 ch 6	802.11 voie 6
802.11 ch 11	802.11 voie 11

NOTE Les numéros de voie montrés sont ceux de l'IEEE 802.11 et de l'IEEE 802.15.4, au lieu de ceux de la présente norme.

Figure 63 – Exemple d'allocation d'intervalles de temps pour le saut de voie discrétisé

La partie inférieure de la Figure 63 montre que le modèle de saut de voie peut être utilisé de façon répétitive au fil du temps. L'utilisation de la taille de supertrame et de l'intervalle de temps par la DLE n'est pas nécessairement liée à des constructions de DL cycliques inférieures telles que les modèles de saut de voies ou les intervalles d'alignement d'intervalles de temps de 250 ms. (V oir 9.1.9.1.3 pour un débat relatif aux intervalles d'alignement.)

La Figure 64 montre l'utilisation du saut de voie lent. Chaque voie est utilisée sur plusieurs intervalles de temps.



Légende

Anglais	Français
Channels	Voies
Slow hopping period 250 ms	Période de saut lent 250 ms
Elapsed time	Temps écoulé

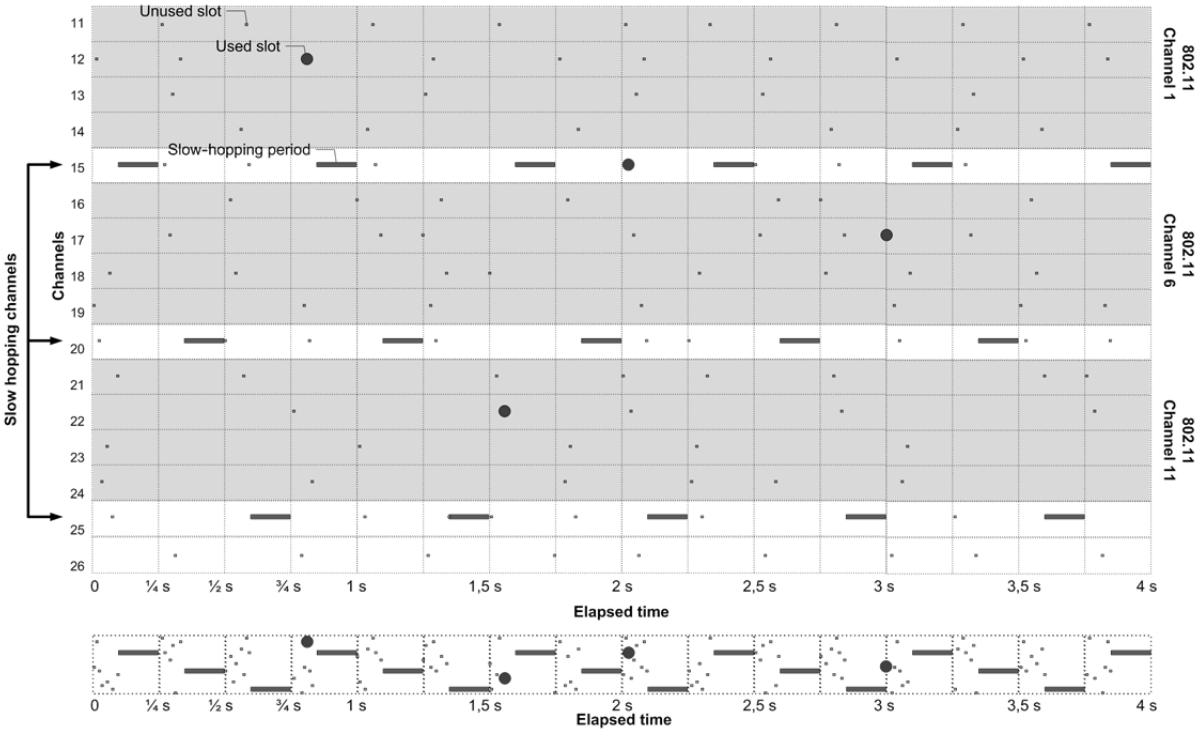
NOTE Les numéros de voie montrés sont ceux de l'IEEE 802.15.4, au lieu de ceux de la présente norme.

Figure 64 – Exemple d'allocation d'intervalles de temps pour le saut de voie lent

Les périodes de saut de voie lent peuvent s'étendre sur un intervalle d'alignement d'intervalles de temps de 250 ms. (Voir 9.1.9.1.3 pour un débat relatif à l'alignement des intervalles de temps.) Dans de tels cas, les périodes de saut de voie lent ne sont pas interrompues par des périodes "idle" c'est-à-dire, si une période de saut de voie lent traverse

la frontière d'un intervalle d'alignement d'intervalles de temps, la radio ne s'éteint pas au cours de la période "idle" autrement exigée.

La Figure 65 montre un système hybride qui combine le saut de voie discrétisé et le saut de voie lent. Dans cet exemple, au sein de chaque intervalle d'alignement de 250 ms, plusieurs intervalles de temps, chacun assigné à une voie différente, sont suivis d'une période saut de voie lent sur une seule voie.



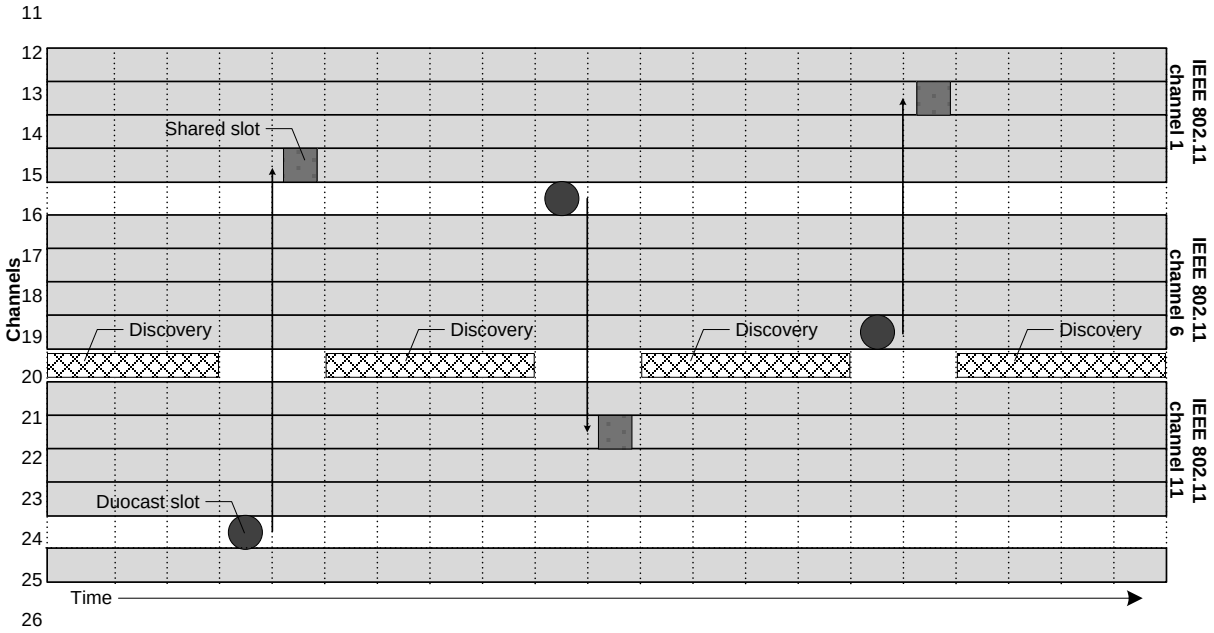
Légende

Anglais	Français
Channels	Voies
Slow hopping channels	Voies de saut lent
Unused slot	Intervalle non utilisé
Used slot	Intervalle utilisé
Slow hopping period	Période de saut lent
Elapsed time	Temps écoulé
802.11 channel 1	802.11 voie 1
802.11 channel 6	802.11 voie 6
802.11 channel 11	802.11 voie 11

NOTE Les numéros de voie montrés sont ceux de l'IEEE 802.11 et de l'IEEE 802.15.4, au lieu de ceux de la présente norme.

Figure 65 – Mode hybride avec saut de voie discrétisé et saut de voie lent

L'ordre dans lequel le saut de voie discrétisé et le saut de voie lent peuvent être combinés est flexible; les périodes de saut de voie lent peuvent ne pas suivre des intervalles de temps de saut de voie discrétisé. Plutôt, les deux peuvent être utilisés dans n'importe quelle combinaison raisonnable. Par exemple, la Figure 66 montre une configuration exemplaire, où une DLE commute entre le saut de voie lent et le saut de voie discrétisé.



Légende

Anglais	Français
Channels	Voies
Shared slot	Intervalle partagé
Discovery	Découverte
Duocast slot	Intervalle en duodiffusion
Time	Temps
IEEE 802.11 channel 1	Voie 1 de l'IEEE 802.11
IEEE 802.11 channel 6	Voie 6 de l'IEEE 802.11
IEEE 802.11 channel 11	Voie 11 de l'IEEE 802.11

NOTE Les numéros de voie montrés sont ceux de l'IEEE 802.11 et de l'IEEE 802.15.4, au lieu de ceux de la présente norme.

Figure 66 – Combinaison de saut de voie lent et de saut de voie discrétisé

Dans l'exemple de la Figure 66, le saut de voie discrétisé est utilisé lorsque des intervalles de temps de communication en diffusion/multidiffusion, duodiffusion/N-diffusion ou à base de conflits sont explicitement alloués. Lorsqu'une DLE n'a pas une allocation d'intervalle de temps, elle se met à l'écoute sur la voie 20, qui facilite la découverte de voisins. (Voir 9.1.9.4.7 pour un débat relatif à la duodiffusion/N-diffusion).

9.1.8 Supertrames

9.1.8.1 Généralités

Une supertrame est une séquence répétitive d'intervalles de temps. Le nombre d'intervalles de temps dans chaque cycle de supertrame (sa taille) et la durée de chacun de ces intervalles de temps déterminent la période du cycle de supertrame. Cela établit la structure du programme de communication pour les DLE qui utilisent la supertrame. Par exemple, une supertrame qui accomplit un cycle toutes les 500 ms permettra à chaque DLE qui utilise un seul intervalle de temps dans la supertrame de communiquer toutes les 500 ms.

Lorsqu'une supertrame est créée, il lui est donné un ID de supertrame. La Figure 67 montre comment des DLE peuvent communiquer dans un exemple de supertrame à trois intervalles de temps.

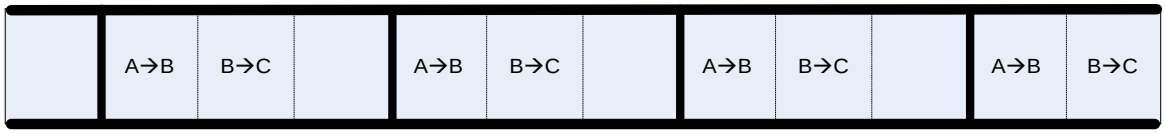
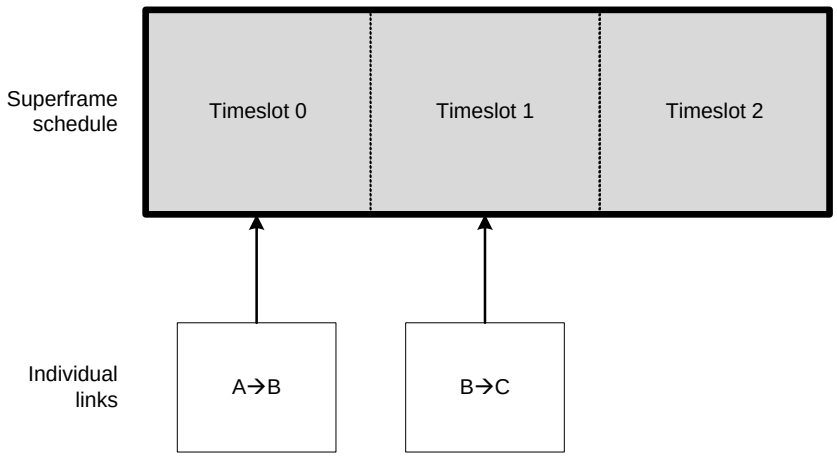


Figure 67 – Exemple de supertrame à trois intervalles de temps et comment elle se répète

A la Figure 67, la DLE A peut communiquer avec la DLE B pendant le premier intervalle de temps de chaque cycle de supertrame. La DLE B peut communiquer avec la DLE C pendant le deuxième intervalle de temps de chaque cycle. Le troisième intervalle de temps de chaque cycle est non assigné ("idle"). Le cycle se répète tous les trois intervalles de temps.

La Figure 67 montre des cycles de temporisation et des liaisons de communication au sein de la même structure, qui est une vue conceptuelle. Les cycles de supertrame et les liaisons de communication sont représentés sous la forme d'objets configurables distincts, mais connexes, au sein de la DLE. La Figure 68 montre cette structure de données, clarifiant la distinction entre supertrames et liaisons, pour la même supertrame à trois intervalles qu'à la Figure 67.



Légende

Anglais	Français
Superframe schedule	Programme de supertrames
Timeslot 0	Timeslot 0 (intervalle de temps 0)
Timeslot 1	Timeslot 1 (intervalle de temps 1)
Timeslot 2	Timeslot 2 (intervalle de temps 2)
Individual links	Liaisons individuelles

Figure 68 – Supertrames et liaisons

Comme montré à la Figure 68, les supertrames se réfèrent à un ensemble d'intervalles de temps. Les liaisons se réfèrent à l'utilisation des intervalles de temps de supertrame pour la communication entre une paire spécifique de DLE. Un intervalle de temps est une période de temps. Une supertrame est un programme cyclique d'intervalles de temps et de voies associées. Une liaison décrit une activité spécifique qui se répète au sein d'un programme cyclique de supertrames.

Le gestionnaire de système configure des jeux concordants de liaisons parmi un ensemble de DLE qui communiquent les unes avec les autres. Par exemple, une liaison sur la DLE A peut être configurée pour émettre des DPDU Data vers la DLE B sur un cycle particulier de supertrame. Sur le même cycle, il convient que la DLE B ait une liaison qui est configurée pour être à l'écoute afin de détecter une DPDU Data entrante. Ces deux liaisons sont

concordantes en ce sens que le gestionnaire de système a configuré ces deux opérations connexes de façon qu'elles se produisent simultanément sur la même voie.

Plusieurs paramètres de performance sont déterminés par la période de supertrame et par la manière dont les liaisons sont assignées aux supertrames. En général, les supertrames de plus courte période conduisent à une latence plus faible des DPDU Data et à une largeur de bande numérique accrue, aux dépens de la consommation accrue d'énergie et d'une allocation plus concentrée de la largeur de bande numérique. Les supertrames de plus longue période conduisent généralement à une latence plus élevée et à une largeur de bande numérique plus petite, mais avec une consommation d'énergie réduite et une allocation moins concentrée de largeur de bande numérique. Il convient d'envisager soigneusement ces compromis pour déterminer la période de supertrame et la densité de liaison au sein d'une supertrame.

Un DLE donnée peut être configurée pour utiliser simultanément plusieurs supertrames de différentes tailles. Une liaison avec un seul intervalle de temps au sein d'une supertrame de la longueur L intervalles de temps se répète deux fois plus fréquemment qu'une semblable liaison à un seul intervalle de temps au sein d'une supertrame de longueur $2 \times L$ intervalles de temps, permettant ainsi la prise en charge de deux fois la sortie par seconde.

Une DLE peut utiliser simultanément plus d'une supertrame. En outre, les DLE dans un sous-réseau D n'ont pas toutes besoin de participer à chaque supertrame. En configurant une DLE pour participer à plusieurs supertrames simultanées de longueurs différentes, il est possible d'établir plusieurs programmes de communication avec des périodes inégales qui opèrent toutes simultanément.

Les supertrames sont numérotées pour l'identification, mais la portée de ces numéros de supertrame est limitée à la DLE où la supertrame est utilisée. Sachant que la portée d'un numéro de supertrame est une seule DLE, une DLE voisine peut utiliser le même numéro de supertrame pour un besoin complètement différent. Les supertrames peuvent être ajoutées, retirées, activées et désactivées pendant que le sous-réseau D fonctionne.

La Figure 69 montre comment des intervalles de temps dans des supertrames différentes sont alignés, bien que les supertrames puissent accomplir un cycle à des fréquences indépendantes.

Légende

Anglais	Français
Superframe 0 5 slots	Supertrame 0 Cinq intervalles
Superframe 1 3 slots	Supertrame 1 Trois intervalles

Figure 69 – Plusieurs supertrames avec intervalles de temps alignés

NOTE L'alignement d'intervalle de temps est un résultat de la définition de tous les intervalles de temps comme ayant des durées identiques et du réalignement des intervalles de temps au temps TAI toutes les 250 ms (voir 9.1.9.1.3). Les supertrames avec des intervalles de temps de durées différentes sont utilisables simultanément au sein d'un sous-réseau D. Cependant, les concepteurs de la présente norme n'ont considéré que des configurations dans lesquelles une seule durée d'intervalle de temps est utilisée lors du fonctionnement d'un sous-réseau D donné (modifiable à la réinitialisation du sous-réseau D).

Une DLE avec plusieurs liaisons dans un intervalle de temps peut rencontrer des collisions de liaison lorsque deux liaisons ou plus de deux liaisons coïncident. Pour adresser de telles

situations, chaque liaison a une priorité assignée. Une priorité de liaison ayant un numéro plus élevé signifie que la liaison a la préséance sur une liaison ayant une priorité de numéro plus faible. En cas de collision de liaison, la liaison avec la priorité de numéro plus élevé est utilisée. Si deux liaisons ont la même priorité, une priorité de supertrame est utilisée. Voir 9.1.8.5.

En plus de la priorité de liaison, chaque DPDU Data a une priorité assignée par sa DLE d'origine. Une fois qu'une liaison est sélectionnée sur la base de la priorité de liaison, la priorité de la DPDU Data est utilisée pour pondérer l'importance relative de toutes les DPDU Data placées en file d'attente qui peuvent utiliser la liaison.

9.1.8.2 Repli exponentiel

Des liaisons peuvent être partagées ou dédiées. Du côté récepteur, il n'y a aucune différence structurelle entre liaisons partagées vs. dédiées. Du côté initiateur de transaction, un bit de repli exponentiel dans la spécification de la configuration de liaison indique si la liaison est partagée ou dédiée. Si une liaison est partagée, comme l'indique le fait que le bit de repli exponentiel de la liaison soit mis (à 1), l'initiateur de transaction doit utiliser le repli exponentiel pour des répétitions de tentative utilisant cette liaison. Si une liaison est dédiée, comme l'indique le fait que le bit de repli exponentiel de la liaison soit réinitialisé (à 0), l'initiateur de transaction ne doit pas utiliser le repli exponentiel dans la liaison en question.

Il est possible, et parfois raisonnable, pour un gestionnaire de système de configurer plusieurs DLE pour qu'elles émettent en même temps et sur la même voie radio, sans positionner un bit de repli exponentiel dans les liaisons applicables. Le terme "liaison dédiée" est utilisé simplement pour indiquer que le repli exponentiel n'est pas appliqué à de telles liaisons.

Le repli exponentiel doit être appliqué lorsque, et seulement lorsque, une DLE émet une DPDU Data en monodiffusion sur une liaison partagée et ne reçoit pas une DPDU ACK/NAK exempte d'erreur, ce qui implique une possible collision. Une émission en monodiffusion qui est prématurément arrêtée en raison de la détection de CCA doit être traitée comme étant l'équivalent d'une émission infructueuse dans le contexte du repli exponentiel. Le repli exponentiel vise à résoudre de telles collisions. Le repli exponentiel doit opérer sur une base voisin par voisin, appliquée à toutes les DPDU Data dans la file d'attente de message adressée au voisin en question, quelle que soit la priorité de la DPDU Data.

Pour chaque voisin, la DLE maintient un exposant de repli et un compteur de repli, appelés BackoffExponent[Neighbor] et BackoffCounter[Neighbor] dans la présente norme. BackoffExponent[ ] et BackoffCounter[ ] sont des parties internes à une mise en œuvre qui sont inaccessibles, et donc ne sont pas incluses dans le modèle d'objet de DL.

BackoffExponent[Neighbor] et BackoffCounter[Neighbor] sont mis à zéro chaque fois qu'une DPDU ACK/NAK est reçue pour une DPDU Data en monodiffusion qui avait été envoyée vers un voisin particulier dans une liaison partagée.

Une valeur de BackoffCounter[Neighbor] égale à zéro permet à la DLE d'envoyer une DPDU Data à la prochaine occasion d'émission de liaison partagée. A la suite d'une émission non réussie vers le voisin dans une liaison partagée, si la valeur courante de BackoffExponent[Neighbor] est inférieure à dlmo.MaxBackoffExp, la DLE incrémente BackoffExponent[Neighbor] et règle BackoffCounter[Neighbor] en sélectionnant une valeur uniformément dans l'intervalle $0..2^{(\text{BackoffExponent[Neighbor]})}-1$. Pour chaque occasion d'émission dans une liaison partagée, BackoffCounter[Neighbor] est décrémenté jusqu'à ce qu'il atteigne zéro. Si une occasion d'émission est dans une liaison dédiée (aucun indicateur de repli exponentiel), la DLE peut utiliser la liaison, quelle que soit la valeur de BackoffCounter[Neighbor]. L'attribut dlmo.MaxBackoffExp limite la valeur maximale de BackoffExponent[Neighbor].

Le comportement de répétitions de tentative peut être configuré par le gestionnaire de système. Les attributs de DLMO qui se rapportent aux répétitions de tentative comprennent:

- dlmo.MaxBackoffExp. La valeur maximale pour BackoffExponent[Neighbor].
- dlmo.MaxLifetime et dlmo.Graph.MaxLifetime: durée de vie maximale d'une DPDU Data. Une DPDU Data qui a été transmise doit être supprimée si elle est maintenue dans une file d'attente de message de DLE plus longtemps que MaxLifetime. L'attribut dlmo.MaxLifetime fournit une valeur par défaut pour la DLE. Une valeur non vide pour dlmo.Graph[ ].MaxLifetime spécifie que dlmo.MaxLifetime doit être neutralisé et mis à la valeur spécifiée pour des DPDU Data suivant ce graphe particulier.
- Le fonctionnement du repli exponentiel qui est illustré dans le pseudocode suivant:

```
// Pour chaque voisin, indépendamment
BExp[Nei] = 0; // BackoffExponent[Neighbor] en texte
BCnt[Nei] = 0; // BackoffCounter[Neighbor] en texte
Pour chaque intervalle de temps (
Si (la liaison d'émission et la DPDU Data correspondent) (// Voir 9.1.8.5
   Si (liaison de repli non exponentielle) (
      // Liaison dédiée
      Tentative d'émission de DPDU Data utilisant une liaison;
      Si (l'émission a réussi) supprimer la DPDU Data de la file d'attente;
      )
   Autre (
      // liaison partagée
      Si (BCnt[Nei] > 0) BCnt[Nei]--
      Autre (
         Tentative d'émission de DPDU Data en liaison;
         Si (l'émission a réussi) (
            Supprimer la DPDU Data de la file d'attente;
            BExp[Nei]=0;
            BCnt[Nei]=0;
            )
         Autre (
            // L'émission a échoué; repli exponentiel
            Si (BExp[Nei] < MaxBackoffExp) BExp[Nei]++;
             BCnt[Nei] = Random (0, 2^(BExp[Nei]-1));
            )
         )
      )
   )
Delete all messages beyond MaxLifetime;
If (no queued Data DPDU for neighbor) (BCnt[Nei]=0; BExp[Nei]=0;)
```

NOTE Comme décrit en 9.1.8.5, il est possible qu'une liaison soit configurée comme liaison d'émission/réception (T/R), qui est une représentation comprimée d'une liaison d'émission et liaison de réception appariées. Logiquement, les liaisons T/R sont traitées comme étant deux liaisons indépendantes.

9.1.8.3 Utilisation de voie de supertrame

Des intervalles de temps dans une supertrame sont associés à un modèle de saut de voie lent ou discrétisé, ainsi qu'à un décalage dans le modèle en question.

De la perspective de chaque DLE utilisant une supertrame, il y a un décalage de modèle de saut de voie de ligne de base, qui peut varier d'une DLE à une autre DLE et qui peut être neutralisé avec un décalage alternatif appliqué à une liaison ou à un ensemble de liaisons au sein d'une supertrame.

Une transaction D en monodiffusion donnée se produit sur une seule voie, avec la DPDU Data et la/les DPDU ACK/NAK DPDU toutes émises sur la même voie.

Une supertrame n'est pas limitée à une seule voie à la fois; au contraire, une supertrame a une structure bidimensionnelle indiquant le temps et la voie, comme cela avait été précédemment montré à la Figure 62 (voir 9.1.7.2.5).

La Figure 62 montre une supertrame, avec le temps sur l'axe horizontal et la voie sur l'axe vertical. La supertrame recouvre toutes les voies sur la longueur (durée) de cette supertrame. Comme montré à la Figure 62, seize DLE peuvent utiliser seize décalages différents à partir

du modèle de saut de voie A; la supertrame englobe toutes les assignations de voie pour l'ensemble de tous les intervalles de temps de supertrame.

Le décalage de voie par défaut peut être différent pour l'émission vs. la réception et peut varier par liaison.

La période du modèle de saut de voie n'est pas nécessairement liée à la longueur de la supertrame. En se référant à la Figure 62, une supertrame pourrait être configurée comme ayant une longueur de 25 intervalles de temps, bien que le modèle de saut de voie A n'ait qu'une longueur de 16 sauts.

Pour la diversité de fréquence, la longueur de supertrame et la taille de modèle de saut de voie peuvent être configurées pour être des nombres premiers l'une par rapport à l'autre, c'est-à-dire, sans facteurs communs. Comme contre-exemple, considérer une configuration dans laquelle la longueur de supertrame est 25 intervalles de temps, avec un modèle de saut de voie se répétant sur un cycle de 15 voies, conduisant à un programme de supertrame où seules 3 des 15 voies disponibles sont utilisées. Un tel plan peut provoquer des problèmes de réglementation dans les situations où l'utilisation de toutes les voies est exigée par chaque appareil.

9.1.8.4 Organisation des supertrames

9.1.8.4.1 Généralités

Les deux types généraux de supertrame sont pris en charge:

- Saut de voie discrétisé, qui fait une utilisation optimale de la largeur de bande numérique disponible et prend en charge les routeurs alimentés par batteries.
- Saut de voie lent, destiné à des routeurs avec de l'énergie disponible pour faire fonctionner leurs récepteurs sans interruption pendant une période donnée. Le saut de voie lent permet à des DLE voisines de fonctionner avec exigences relatives à la synchronisation du temps moins contraignantes, en particulier au cours du processus de découverte de voisins.

Des configurations hybrides peuvent être arrangées en combinant les supertrames, par exemple, une discrétisée et une lente. Les sauts de voie discrétisé et lent seront débattus séparément, suivi d'un certain nombre d'exemples de configurations hybrides.

NOTE Sur le marché, le saut de voie lent est parfois appelé CSMA, et le saut de voie discrétisé AMRT. Ces termes ne sont pas utilisés dans la présente norme, excepté dans la mesure où le CSMA/CA est pris en charge par la norme dans un sens littéral. (voir 9.1.9.4.8.) Le saut de voie lent est construit sur la base de l'AMRT, le saut de voie discrétisé inclut des aspects du CSMA. Le concepteur de solution est libre de mélanger les approches.

9.1.8.4.2 Portée de supertrame

Les supertrames sont généralement débattues comme étant des abstractions qui enjambent plusieurs DLE. Néanmoins, alors que la supertrame peut être conceptualisée au niveau du sous-réseau D, la portée de la structure de données de supertrame est limitée à chaque DLE. Une supertrame est instanciée comme une structure de données sur une seule DLE qui commande indépendamment son diagramme d'états de DLE. Les définitions de supertrame d'une DLE ont besoin de se rapporter à celles de ses voisins afin que les DLE communiquent en même temps. Des définitions de supertrame au sein de chaque DLE sont numérotées, mais cette numérotation est nécessaire seulement au DMAP pour des opérations de lecture/écriture de table et à d'autres objets et attributs au sein de la DLE qui se réfèrent à la supertrame.

Une supertrame peut être contrastée avec une portée de graphe de routage. Un ID de graphe, à la différence d'un ID de supertrame, est transporté dans un en-tête DROUT d'une DPDU Data, et un ID de graphe doit être cohérent et unique dans toutes les DLE qui utilisent le graphe. Aucune contrainte de ce type ne s'applique à une supertrame.

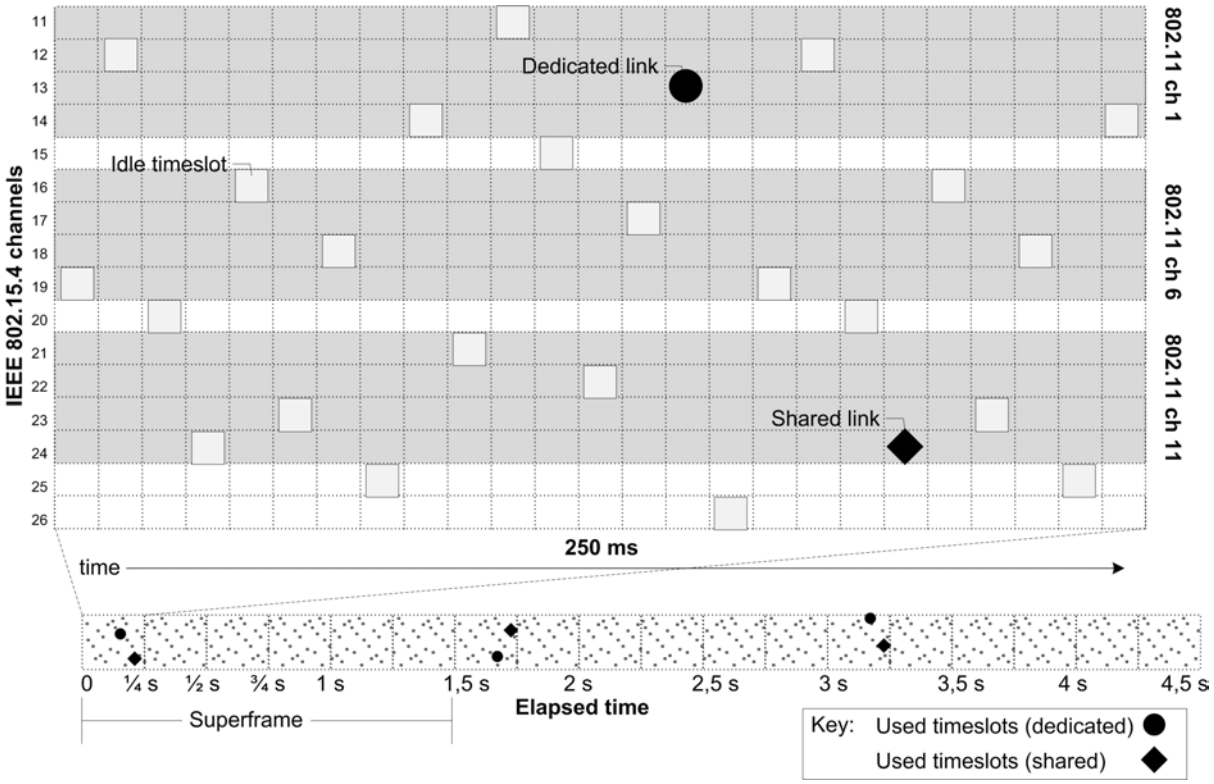
9.1.8.4.3 Blocs de capacité à base de conflits

Les routeurs alimentés par secteur peuvent raisonnablement actionner leurs récepteurs en permanence. En tant que supertrame ayant la priorité la plus basse, un tel routeur peut prendre en charge une supertrame constituée, partiellement ou entièrement, de liaisons de réception. Les voisins du routeur peuvent maintenir les liaisons d'émission correspondantes partagées vers le routeur. Une telle configuration conduit à des blocs de largeur de bande numérique à base de conflits disponibles pour les voisins des routeurs. Le saut de voie lent ou le saut de voie discrétisé peut être utilisé dans une supertrame de ce type. Le décalage de saut de voie peut être sélectionné pour éviter des collisions entre les liaisons dédiées vs. un inventaire général des liaisons partagées.

9.1.8.4.4 Saut de voie discrétisé

Le saut de voie discrétisé utilise des intervalles de temps de supertrame de saut de voie d'égale durée. Chaque intervalle de temps de supertrame utilise une voie radio différente dans un modèle de saut. Dans le saut de voie discrétisé, chaque intervalle de temps de supertrame est prévu pour prendre en charge une seule transaction D, y compris une DPDU Data et sa/ses DPDU ACK/NAK.

La Figure 70 montre une supertrame de saut de voie discrétisé à partir de la perspective d'une DLE, qui peut être un routeur.



Légende

Anglais	Français
IEEE 802.15.4 channels	Voies IEEE 802.15:4
Dedicated link	Liaison dédiée
Idle timeslot	Intervalle de temps inactive
Shared link	Liaison partagée
Time	Temps
Superframe	Supertrame
Elapsed time	Temps écoulé

Anglais	Français
Key	Légende
Used timeslots (dedicated)	Intervalles de temps utilisés (dédiés)
Used timeslots (shared)	Intervalles de temps utilisés (partagés)
802.11 ch 1	802.11 voie 1
802.11 ch 6	802.11 voie 6
802.11 ch 11	802.11 voie 11

NOTE Les numéros de voie montrés sont ceux de l'IEEE 802.11 et de l'IEEE 802.15.4, au lieu de ceux de la présente norme.

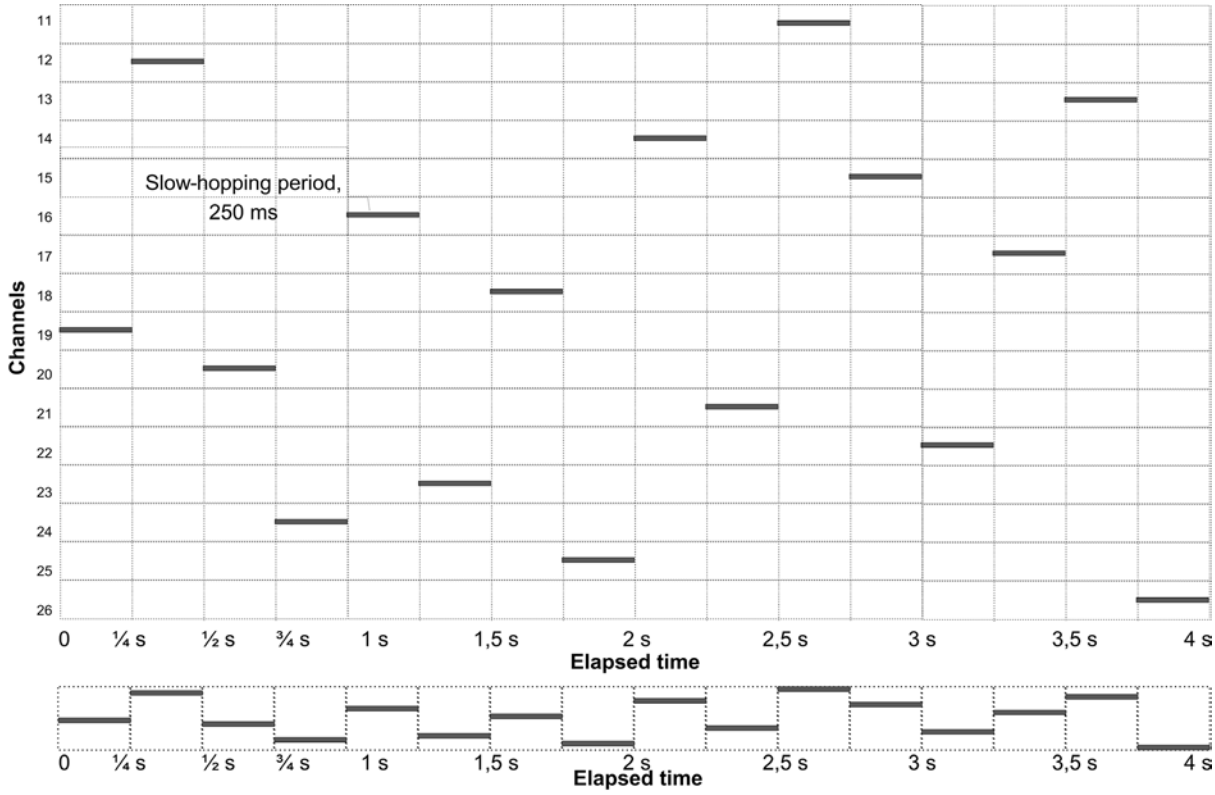
Figure 70 – Exemple de supertrame pour saut de voie discrétisé

Dans cet exemple, la supertrame a une longueur de 1,5 s. Des intervalles de temps avec des assignations de liaison sont représentés avec des cercles (liaisons dédiées) et des losanges (liaisons partagées). Comme le montre la Figure 70, à partir de la perspective d'une seule DLE, de nombreux intervalles de temps de supertrame pourraient être laissés "idle". Les intervalles de temps avec assignations de liaison se répètent à un intervalle fixe défini par la longueur de supertrame.

9.1.8.4.5 Saut de voie lent

Dans le saut de voie lent, un ensemble d'intervalles de temps de supertrame contigus est groupé sur une seule voie radio. Chacun de tels ensembles d'intervalles de temps de supertrame est traité comme une seule période de saut de voie lent.

La Figure 71 montre le saut de voie lent.



Légende

Anglais	Français
Channels	Voies
Slow hopping period 250 ms	Période de saut lent 250 ms

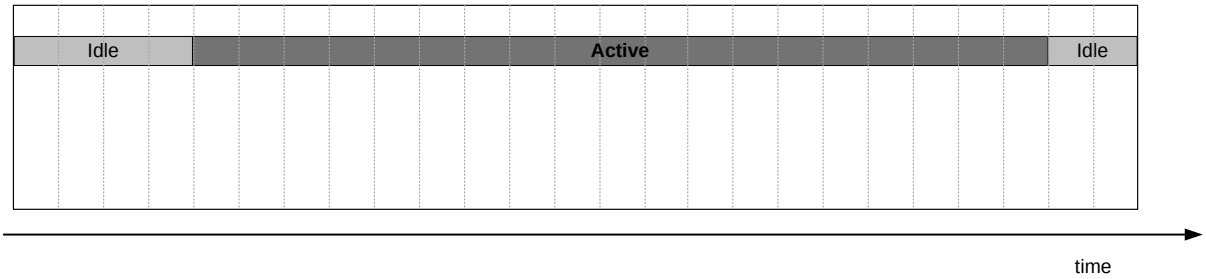
Anglais	Français
Elapsed time	Temps écoulé

NOTE Les numéros de voie montrés sont ceux de l'IEEE 802.15.4, au lieu de ceux de la présente norme.

Figure 71 – Exemple de supertrame pour saut de voie lent

Les intervalles de temps dans une supertrame de saut de voie lent sont généralement partagés, fournissant une largeur de bande de voie à base de conflits instantanée, à la demande, aux voisins immédiats d'un routeur.

La Figure 72 montre les principales composantes d'une supertrame de saut de voie lent.



Légende

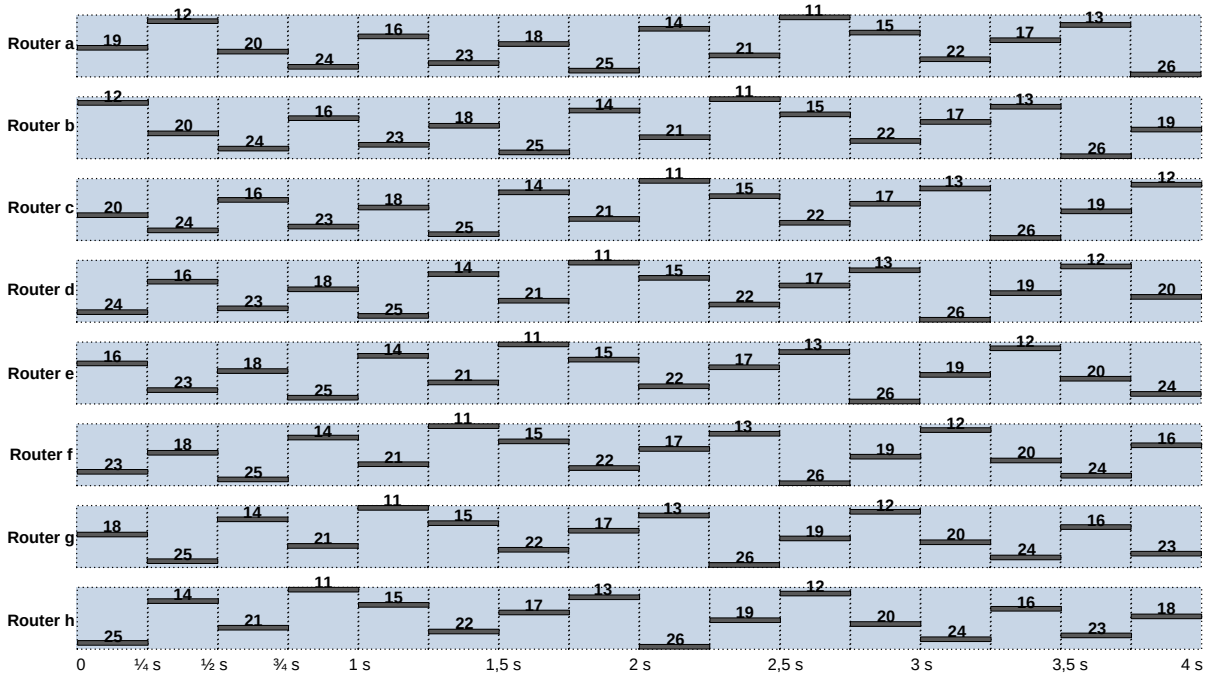
Anglais	Français
Idle	Inactive
Active	Active
Time	Temps

Figure 72 – Composantes d'une supertrame de saut de voie lent

Une période de saut de voie lent de ligne de base a une durée fixe, avec un nombre fixe d'intervalles de temps "idle" (qui peuvent être zéro) au début et à la fin du saut. L'exemple d'une période de saut de voie lent montré à la Figure 72 est composé de 25 intervalles de temps, y compris quatre intervalles de temps inactifs au début, dix-neuf intervalles de temps actifs au milieu, et deux intervalles de temps inactifs à la fin. Les intervalles de temps inactifs visent à prendre en charge des configurations hybrides où des supertrames de saut de voie lent sont appariées à des supertrames de saut de voie discrétisé, avec les intervalles de temps de saut de voie discrétisé programmés pour être utilisés pendant les périodes inactives de la supertrame de saut de voie lent.

NOTE 1 Les périodes inactives telles que décrites ici sont configurables en appariant les phases de supertrame et de saut de voie, et en définissant les liaisons qui concordent avec la plage active souhaitée.

Les sauts ne sont pas nécessairement effectués ensemble, par tous les routeurs d'une zone commune. La Figure 73 montre le nombre de routeurs disjoints les uns des autres auxquels il est possible d'assigner des modèles de sauts lents, évitant ainsi les collisions. Cela n'implique en aucune manière que tous les routeurs d'un système utilisent des voies de communication disjointes.



Légende

Anglais	Français
Router	Routeur

NOTE Les numéros de voie montrés sont ceux de l'IEEE 802.15.4, au lieu de ceux de la présente norme.

Figure 73 – Exemple de configuration pour éviter les collisions entre routeurs

Chaque routeur peut utiliser un décalage différent dans le modèle de saut de voie. Avec un modèle de saut de 16 voies, chacun d'un maximum de 16 routeurs peut être configuré avec un décalage différent dans le modèle et, donc, deux routeurs quelconques n'utilisent pas la même voie en même temps.

NOTE 2 Bien que la Figure 73 prétende montrer comment il est possible d'éviter les collisions par des assignations disjointes de modèles de sauts de voies, un système réel exige au moins un modèle de saut de voie partagé via lequel les routeurs peuvent communiquer entre eux.

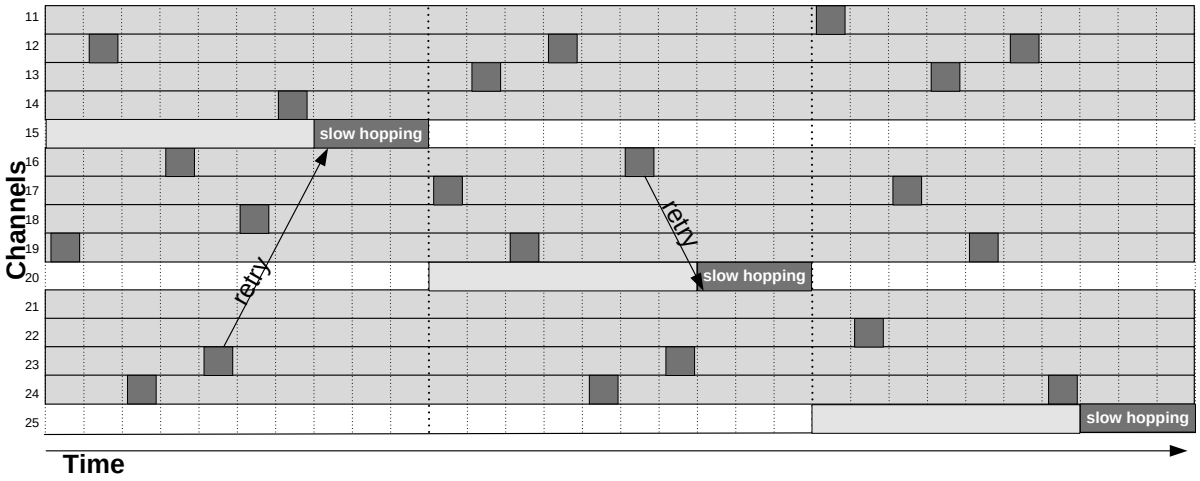
Voir 9.4.3.5.5 pour plus de détails relatifs au saut de voie lent.

9.1.8.4.6 Configurations hybrides de saut de voie

Les configurations hybrides peuvent utiliser des combinaisons de supertrames de saut de voie discrétisé et de saut de voie lent.

Des configurations hybrides sont habituellement arrangées de sorte que des liaisons de saut de voie discrétisé soient allouées pour la messagerie périodique programmée. Cela peut laisser des blocs de capacité de saut de voie lent légèrement utilisés, disponibles sur une base de conflits, pour des utilisations moins prévisibles telles que des alarmes et des répétitions de tentative.

L'exemple à la Figure 74 montre comment les supertrames de saut de voie discrétisé et de saut de voie lent peuvent être combinées.



Légende

Anglais	Français
Channels	Voies
slow hopping	saut lent
retry	répétition de tentative
Time	Temps

NOTE Les numéros de voie montrés sont ceux de l'IEEE 802.15.4, au lieu de ceux de la présente norme.

Figure 74 – Configuration hybride

A la Figure 74, le saut de voie discrétisé a été couché sur un arrière-plan de saut de voie lent. Les périodes de saut lent remplissent le temps entre les ensembles périodiques d'intervalles de temps dédiés de supertrames.

Dans cette configuration, si une émission tentée dans un intervalle de temps dédié échoue, la période de saut de voie lent qui suit peut être utilisée pour répéter la tentative d'émission. A la Figure 74, deux des intervalles de temps de supertrame sont montrés avec des répétitions de tentative sur des voies différentes au cours de la période suivante de saut de voie lent.

9.1.8.4.7 Supertrames et gestion de spectre

Le terme gestion de spectre se réfère à la capacité du gestionnaire de système de configurer un sous-réseau D pour bloquer le fonctionnement de voies non désirées dans un sous-réseau D.

Chaque supertrame inclut une carte de voies, appelée Superframe[ ].ChMap, qui est un masque de bits des voies qui doivent être incluses et exclues dans la séquence de saut qui est référencée par la supertrame. Les voies exclues ont l'effet de raccourcir la séquence de saut.

Par exemple, une carte de voies de supertrame qui inclut les voies 11..25 et exclut la voie 26 a l'effet de raccourcir la séquence de saut en retirant la voie 26 de la séquence de saut. Plus généralement, le gestionnaire de système peut éliminer n'importe quel ensemble de voies des séquences de saut qui sont référencées par des supertrames dans un sous-réseau D, avec le résultat de retirer ces voies du fonctionnement.

Il y a également une carte de voies dans l'attribut DeviceCapability qui est rapporté au gestionnaire de système lorsque la DLE rejoint le sous-réseau D. La carte de voies de DeviceCapability ne raccourcit aucune séquence de saut de voie utilisée par la DLE, mais est plutôt un signal donné au gestionnaire de système lui indiquant que, pour des raisons de réglementation, toute liaison utilisant l'une des voies exclues sera traitée comme étant "idle".

Le gestionnaire de système peut également bloquer l'utilisation de certaines voies par l'intermédiaire de l'attribut dlmo.IdleChannels. A la différence de la carte de voies dans les supertrames, dlmo.IdleChannels ne provoque pas le raccourcissement des séquences de saut. A la place, il amène les liaisons sur les voies spécifiées à être traitées comme étant "idle".L'attribut dlmo.IdleChannels vise à fournir au gestionnaire de système un moyen rapide de désactiver certaines voies d'une manière qui n'exige pas la coordination à l'échelle du sous-réseau D des séquences révisées de saut.

Le diagnostic spécifique à une voie, tel que décrit en 9.4.2.27, fournit au gestionnaire de système les informations pour prendre en charge la gestion de spectre.

9.1.8.5 Fonctionnement de la file d'attente de messages de DLE

Les routeurs de DL conformes à la présente norme doivent prendre en charge une file d'attente de messages de DLE. Cette file d'attente de messages a un certain nombre d'attributs qui peuvent être configurés par le gestionnaire de système, affectant la façon dont la file d'attente fonctionne.

La norme ne spécifie pas généralement les mécanismes internes de la DLE. Cependant, à un degré limité, la file d'attente de messages de DLE est spécifiée par la norme. Le gestionnaire de système peut configurer la file d'attente de messages de DLE pour réaliser les objectifs particuliers de qualité de service, et un modèle limité du comportement de file d'attente de messages est implicite dans les variantes de configurations fournies au gestionnaire de système.

Lorsqu'une DLE reçoit une DPDU Data en monodiffusion issue du voisin, elle évalue d'abord s'il convient que la DSDU soit passée à la NL ou soit transmise à une autre DLE à travers la DL, comme décrit en 9.3.3.6.

Si la DPDU Data a besoin d'être transmise, la DLE évalue s'il convient que la DPDU Data soit acceptée ou acquittée négativement (NAK), en se basant en partie sur la capacité disponible de la file d'attente de messages de DLE. Les DPDU Data doivent être acquittées négativement (NAK) si la file d'attente de messages de DLE a épuisé la capacité pour les DPDU Data du type en question.

La DL rapporte sa capacité de file d'attente de transmission au gestionnaire de système lorsque la DLE rejoint le sous-réseau D, par le biais de l'attribut dlmo.DeviceCapability, champ QueueCapacity. Cette capacité de file d'attente rapportée à l'extérieur n'inclut pas les parties de la file d'attente qui sont réservées à l'usage interne de la DLE. Par exemple, la file d'attente de messages de DLE dans une DLE particulière pourrait rapporter qu'elle a une capacité de file d'attente pour cinq DPDU Data. Le gestionnaire de système est alors capable de configurer ces cinq positions dans la file d'attente. Cette capacité nominale se réfère seulement à une partie de la file d'attente de messages qui est exclusivement utilisée pour acheminer des DPDU Data par la DLE. Dans la pratique, la DLE réelle a un espace supplémentaire de capacité de file d'attente de messages qu'elle ne rapporte pas au gestionnaire de système, car la DLE a également besoin de traiter des DPDU Data pour son propre compte. La capacité non rapportée de tampon de messages, pour le propre usage de la DLE, est considérée comme étant une question interne de DLE et n'est pas allouée par le gestionnaire de système. Le gestionnaire de système suppose que la DLE a la capacité suffisante de file d'attente de messages pour son propre usage lorsque des contrats sont octroyés.

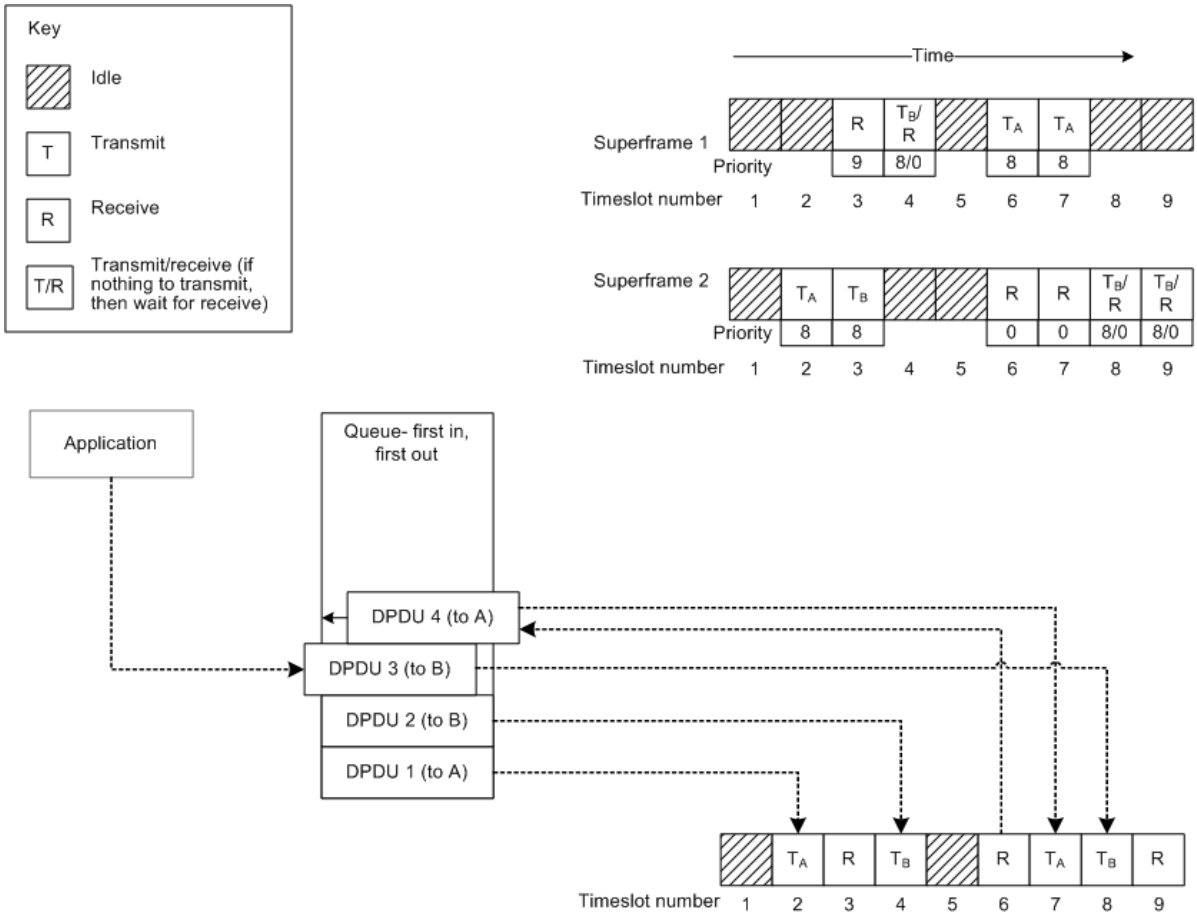
Considérons l'exemple d'un routeur de terrain avec une capacité rapportée de tampon de messages de DLE de huit DPDU Data. La capacité réelle de tampon peut être de douze DPDU Data, auquel cas la différence entre la capacité rapportée et la capacité réelle (quatre DPDU Data) est pour le propre usage de la DLE. Dans cet exemple, le gestionnaire de système pourrait raisonnablement configurer la capacité de tampon nominale de DLE comme suit:

- Pas plus de trois des huit tampons seront utilisés pour transmettre des DPDU Data avec la priorité ≤ 2.
- Pas plus de cinq des huit tampons peuvent utilisés pour transmettre des DPDU Data avec la priorité ≤ 5.

Voir 9.4.2.26 pour un débat plus poussé relatif à la capacité de tampon de file d'attente et aux niveaux de priorité.

En outre, pour un degré de commande plus fin, le gestionnaire de système peut spécifier un certain nombre de tampons exclusivement pour transmettre des DPDU Data qui sont en cours d'acheminement le long du graphe spécifique, comme décrit en 9.4.3.7.

La Figure 75 fournit une vue d'ensemble de la façon dont les liaisons interagissent avec une file d'attente implicite de messages de DLE au sein d'une DLE.



Légende

Anglais	Français
Key	Légende
Idle	Inactif
Transmit	Emission
Receive	Réception
Transmit/receive (if nothing to transmit, then wait for receive)	Emission/réception (si rien à émettre, alors attendre de recevoir)
Time	Temps
Superframe 1	Supertrame 1
Priority	Priorité

Anglais	Français
Timeslot number	Numéro d'intervalle de temps
Superframe 2	Supertrame 2
Application	Application
Queue-first in, first out	File d'attente – premier entré premier sorti
DPDU 4 (to A)	DPDU 4 (vers A)
DPDU 3 (to B)	DPDU 3 (vers B)
DPDU 2 (to B)	DPDU 2 (vers B)
DPDU 1 (to A)	DPDU 1 (vers A)

Figure 75 – Allocation d'intervalles et file d'attente de messages

Comme montré à la Figure 75, des Data DPDU sont maintenues dans une file d'attente de messages jusqu'à ce que des liaisons d'émission deviennent disponibles. Des DPDU Data sont placées dans la file d'attente dans l'ordre de leur réception. Généralement, la récupération des DPDU Data dans la file d'attente est premier entré, premier sorti (FIFO).

NOTE Cet exemple simplifié montre seulement une seule adresse de destination pour chaque DPDU Data. Dans la pratique, chaque DPDU Data sur la file d'attente de messages est réellement une candidate pour des liaisons vers plusieurs voisins.

Considérer l'exemple de la DPDU Data 3. La DPDU Data 3 a pour origine une application et pénètre dans la DL par le DLSAP (Figure 53). Lorsque la DPDU Data est traitée par la DLE, elle est placée dans la file d'attente de messages de la DLE. En fonction de la destination finale des DPDU Data, la DLE détermine que la DPDU Data a besoin d'une liaison vers la DLE B pour son prochain saut. Les DPDU Data 1 et 2 sont déjà dans la file d'attente et, donc, la DPDU Data 3 est placée dans la file d'attente derrière elles. Lorsqu'une liaison vers la DLE B devient disponible, la DPDU Data 2 sera envoyée avant la DPDU Data 3.

Chaque DPDU Data dans la file d'attente a une priorité assignée par la DLE émettrice. Ce tutoriel simplifié suppose que toutes les DPDU Data ont une égale priorité. Dans la pratique réelle, la priorité de liaison a la préséance sur la priorité de DPDU Data, en ce sens que des DPDU Data ne sont pas considérées pour l'émission qu'après la sélection d'une liaison d'émission. Une fois qu'une liaison d'émission a été sélectionnée, la DPDU Data dans la file d'attente est sélectionnée d'abord en fonction de la priorité et ensuite en fonction d'une base FIFO si plusieurs DPDU Data candidates ont la même priorité.

La DPDU Data 4 montre un exemple où une DPDU Data est reçue dans un intervalle de temps donné et est disponible pour être transmise dans le prochain intervalle de temps. En général, une DPDU Data reçue dans un intervalle de temps donné (intervalle de temps N) doit être disponible dans la file d'attente pour une transmission dans le prochain intervalle de temps (intervalle de temps N+1). Une exception est autorisée lorsque le modèle d'intervalle de temps par défaut 3 est utilisé (voir Figure 157). Dans ce cas, la transaction D pourrait être inachevée au début du prochain intervalle de temps. Une DPDU Data reçue en commençant dans un intervalle de temps donné (intervalle de temps N), en utilisant le modèle d'intervalle de temps par défaut 3, doit être disponible dans la file d'attente pour une transmission dans l'intervalle de temps suivant le prochain intervalle de temps (intervalle de temps N+2).

A la Figure 75, deux supertrames sont montrées, chacune étant associée à une série de liaisons. Par exemple, l'intervalle de temps 3 dans la supertrame 1 est associé à une liaison de réception (R), alors que l'intervalle de temps 3 dans la supertrame 2 est associé à une liaison d'émission vers la DLE B (T_B). Les intervalles de temps inactifs n'ont pas de liaisons définies.

Chaque liaison a une priorité. Les attributs dlmo.LinkPriorityXmit et dlmo.LinkPriorityRcv fournissent les priorités de liaison par défaut, qui peuvent être neutralisées pour n'importe quelle liaison particulière. L'exemple à la Figure 75 montre surtout les priorités par défaut 0 pour les liaisons de réception et 8 pour les liaisons d'émission.

Dans la plupart des cas, il convient que le gestionnaire de système assigne une plus faible priorité aux liaisons de réception (R) qu'aux liaisons d'émission (T), donnant ainsi la préséance pour desservir les DPDU Data sortantes dans sa file d'attente. Cependant, il ne s'agit pas d'une exigence stricte. Par exemple, si un flux entrant à la latence critique est programmé pour un intervalle de temps particulier, le gestionnaire de système peut configurer des liaisons de réception avec une priorité plus élevée dans le cas en question. Comme illustration à la Figure 75, la troisième liaison dans la supertrame 1 a une priorité assignée de 9, donnant à cette liaison de réception particulière une plus haute priorité qu'une liaison de transmission au même moment.

Les intervalles de temps d'émission/réception (T/R) utilisent un format compressé pour combiner une liaison d'émission et une liaison de réception. Logiquement, une liaison de T/R constitue deux liaisons, la partie réception de la liaison ayant une priorité de dlmo.LinkPriorityRcv qui prend une valeur par défaut de zéro. Par exemple, si un intervalle de temps a une liaison T_A/R de haute priorité et une liaison T_B de plus faible priorité, la liaison T_B a une plus haute priorité que la partie R de la liaison T_A/R. Le fonctionnement de ligne de base est qu'une DPDU Data placée en file d'attente pour une émission et une liaison sont appariées au début d'un intervalle de temps, et l'intervalle de temps est assigné à une transaction D qui sera exécutée jusqu'à son terme en fonction de la configuration de la liaison. Au cas où une transaction D serait prématurément arrêtée en raison de la détection CCA d'une activité de voies concurrentes, une mise en œuvre optimisée peut compléter l'intervalle de temps en utilisant une liaison de réception qui est valide pour le même intervalle de temps.

Une fois qu'une DPDU Data placée en file d'attente a été émise et acquittée, la transaction D est réputée avoir réussi et la DPDU Data est supprimée de la file d'attente. L'exemple suivant suppose que toutes les transactions ont réussi.

La boîte en bas à droite de la Figure 75 montre comment les liaisons sont utilisées dans l'exemple ci-après.

- Intervalle de temps 1: La première liaison dans les deux supertrames 1 et 2 est inactive; par conséquent, le premier intervalle de temps est inactif.
- Intervalle de temps 2: La deuxième liaison dans la supertrame 1 est inactive, mais il y a une liaison d'émission pour la DLE A (T_A) dans la supertrame 2. Il y a également une DPDU Data destinée à la DLE A dans la file d'attente; par conséquent, le deuxième intervalle de temps est assigné pour une émission vers la DLE A, et la DPDU Data 1 est envoyée.
- Intervalle de temps 3: La supertrame 1 a une liaison de réception et la supertrame 2 a une liaison d'émission. La liaison dans la supertrame 1 a la préséance en raison de sa priorité, si bien que l'intervalle de temps 3 est utilisé pour se mettre à l'écoute afin de déceler des DPDU Data entrantes. (Comme décrit ci-dessus, les liaisons de réception ont habituellement une priorité assignée inférieure à celle des liaisons d'émission. Cet exemple montre comment un gestionnaire de système peut donner la priorité à une liaison de réception, par exemple, pour desservir un flux entrant.)
- Intervalle de temps 4: La supertrame 1 a une liaison T_B/R, indiquant qu'il convient qu'elle émette s'il y a une DPDU Data destinée à la DLE B dans la file d'attente. La DPDU Data est envoyée.
- Intervalle de temps 5: Les deux supertrames sont inactives dans l'intervalle de temps 5, si bien que l'intervalle de temps est inactif.
- Intervalle de temps 6: La supertrame 1 désigne une ligne d'émission vers la DLE A, mais il n'y a plus de DPDU Data destinée à la DLE A dans la file d'attente. Sachant qu'il n'y a rien d'utile à faire pour la liaison d'émission dans cet intervalle de temps, la liaison dans la supertrame 2 est utilisée. La DLE reçoit des DPDU Data entrantes, détermine que son prochain saut est vers la DLE A, et place la DPDU Data dans la file d'attente.
- Intervalle de temps 7: Maintenant qu'il y a une DPDU Data pour la DLE A dans la file d'attente, la liaison d'émission dans la supertrame 1 obtient la priorité et la DPDU Data

4 est envoyée. Noter que la DPDU Data 3 a été sautée parce qu'aucune liaison T_B n'est encore devenue disponible.

- Intervalle de temps 8: Maintenant qu'une liaison T_B est disponible, la DPDU Data 3 est envoyée.
- Intervalle de temps 9: La liaison T_B/R conduit à un intervalle de temps de réception, car il n'y a pas de DPDU Data destinée à la DLE B dans la file d'attente.

Cet exemple a été simplifié en un certain nombre d'aspects essentiels.

– Comme noté ci-dessus, les priorités des DPDU Data ont été supposées être égales. Dans la pratique, si deux DPDU Data sur une file d'attente de messages concordent toutes les deux avec une liaison donnée, la DPDU Data avec la priorité la plus élevée est émise. La position dans la file d'attente FIFO n'est pertinente que pour des DPDU Data d'égale priorité.

– Si des DPDU Data en monodiffusion ne reçoivent pas un acquittement, elles restent dans la file d'attente et la stratégie de répétitions de tentative de la DL est appliquée. Voir 9.1.8.2.

– Une liaison peut être configurée pour être utilisée par un graphe particulier. Dans ce cas, les DPDU Data avec cet ID de graphe doivent avoir un accès hiérarchisé par ordre de priorité ou exclusif à cette liaison. Voir 9.4.3.7.

– Une DLE peut être conçue pour sauter des liaisons sur les voies radio avec des antécédents de connectivité inférieure à la normale. Une DLE peut également sauter des liaisons afin de répéter la tentative sur une liaison de remplacement. Voir 9.1.7.2.4.

Le fonctionnement du traitement de file d'attente est illustré dans le pseudocode suivant:

```
Pour chaque intervalle de temps (
   Ordonner les DPDU Data dans la file d'attente par priorité;
   // FIFO dans la priorité

   Ordonner les liaisons par priorité;
   // Traiter les liaisons T/R comme deux liaisons, avec un côté réception
   // de la liaison assignée priorité de liaison dlmo.LinkPriorityRcv;
   // Dans la priorité de liaison, ordonner par priorité de supertrame,
   // puis numéro de supertrame (dans l'ordre décroissant);

   Pour chaque liaison, dans l'ordre de priorité
      Si c'est une liaison reçue (
         Utiliser la liaison reçue;
         Effectué avec intervalle de temps;
         )
      Autre // c'est une liaison émise
         Pour chaque DPDU Data, dans l'ordre de priorité
            Si la liaison correspond à la DPDU Data (
                  Utiliser la liaison;
                  Effectué avec intervalle de temps;
                  )
)
```

9.1.9 Chronométrage de DL

9.1.9.1 Temporisation

9.1.9.1.1 Généralités

La DL propage et utilise le temps atomique international (TAI) pour son fonctionnement interne, et elle fournit également le temps TAI comme service (par le DMAP) aux appareils sans fil conformes à la présente norme.

Les DLE au sein d'un sous-réseau D peuvent être configurées pour suivre à la trace le sens partagé du temps à quelques millisecondes près l'une de l'autre, pour prendre en charge la séquence de comptes-rendus d'événements et autres opérations coordonnées de couche d'application dans la portée d'un sous-réseau D. Le temps synchronisé parmi les voisins immédiats est également essentiel au fonctionnement du protocole sans fil.

Le saut de voie discrétisé exige une synchronisation serrée du temps parmi les voisins immédiats. Cependant, l'horloge interne d'une DLE autre que de routage qui a été déconnectée du fonctionnement du sous-réseau D pendant plus de quelques minutes peut avoir dérivé de dizaines ou de centaines de millisecondes, ou plus, par rapport à l'horloge globale du sous-réseau D. Les configurations de saut de voie lent ou hybride prennent en charge le fonctionnement continu de ces DLE.

9.1.9.1.2 Temps atomique international

Dans la présente norme, le temps est basé sur le temps atomique international (TAI) comme référence temporelle. Voir 5.6.

9.1.9.1.3 Intervalles d'alignement

Dans la présente norme, les incréments TAI d'une seconde sont divisés en intervalles d'alignement de 250 ms (1/4 s), dans lesquels les cycles de 250 ms (2^{-2} s) doivent s'aligner à la seconde TAI nominale, conformément à la Figure 76.

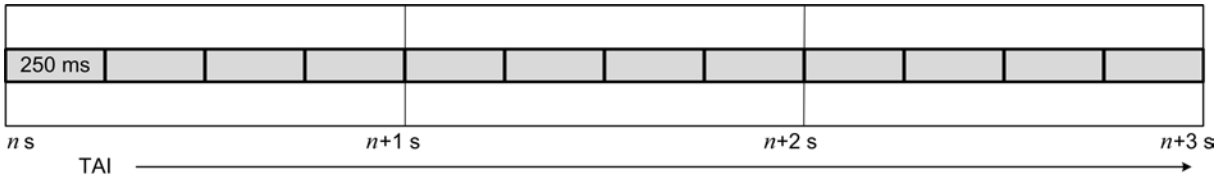


Figure 76 – Intervalles d'alignement de 250 ms

Les boucles de commande continues dans les industries de transformation, qui ont été un centre d'intérêt traditionnel du CE 65 de l'IEC, sont fréquemment basées sur le calcul à période fixe des sorties de commande. Ces boucles de transformation se répètent habituellement à des fréquences de 4 Hz (250 ms), de 1 Hz (1 s) ou de multiples de 4 s ou 5 s. La surveillance de processus est habituellement mappée à cette même structure de 4 Hz (et plus lente).

Les applications avec des fréquences de boucle légèrement plus élevées, comme les boucles de commande de pompage de compresseurs fonctionnant à 12 Hz, peuvent être prises en charge en programmant plusieurs occasions de communications par cycle de 250 ms.

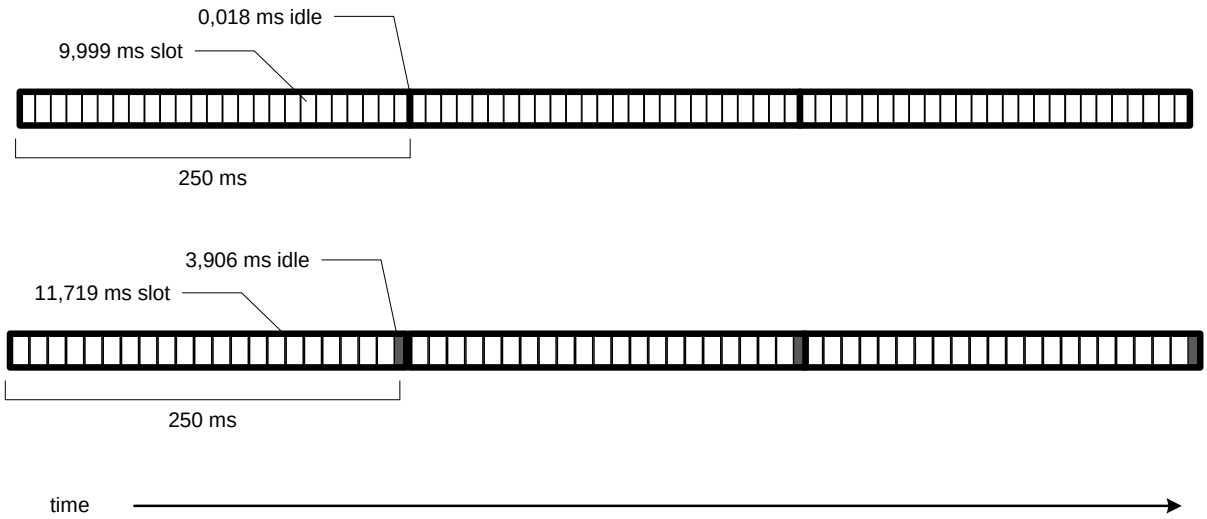
9.1.9.1.4 Durée d'intervalle de temps, alignement d'intervalle de temps et périodes inactives

Dans les intervalles d'alignement de 250 ms, le temps est divisé en intervalles de temps de durée configurable, mais égale. Un intervalle de temps est un intervalle de temps de durée prédéfinie utilisée pour envoyer ou recevoir une DPDU Data et toutes les éventuelles DPDU ACK/NAK correspondantes. Les intervalles de temps sont généralement partagés par au moins une paire de DLE qui communiquent pendant le temps alloué. La DL organise ces intervalles de temps en supertrames, qui sont des ensembles d'intervalles de temps avec une période commune et potentiellement d'autres attributs communs.

Les durées d'intervalle de temps sont configurées au cours de l'établissement du sous-réseau D. Normalement, pendant une période donnée de fonctionnement, tous les intervalles de temps au sein d'un sous-réseau D ont la même durée. La durée d'intervalle de temps est configurée en unités de 2^{-20} s (~0,95 µs).

NOTE 1 La DL lie la durée d'intervalle de temps à des supertrames, et rien dans la norme n'empêche plusieurs supertrames avec des durées différentes d'intervalle de temps d'être actives simultanément au sein d'un sous-réseau D. Cependant, la présente norme ne considère que des configurations dans lesquelles une seule durée d'intervalle de temps est utilisée au sein d'un sous-réseau D donné. Une DLE prenant en charge plusieurs durées d'intervalle de temps simultanément, telle qu'un routeur dorsal ou un pont entre deux sous-réseaux D, peut être modélisée comme contenant plusieurs DLE fonctionnant en parallèle.

Les intervalles de temps s'alignent avec des intervalles d'alignement de 250 ms, mais les durées d'intervalle de temps ne se divisent pas nécessairement de manière uniforme en 250 ms. Par conséquent, le système insère une courte période inactive toutes les 250 ms selon les besoins, réalignant ainsi les intervalles de temps à un cycle de 4 Hz. Cela est montré à la Figure 77 avec deux exemples illustratifs utilisant des durées différentes d'intervalles de temps respectives de 9,999 ms et de 11,719 ms.



Légende

Anglais	Français
0,018 ms idle	Inactif 0,018 ms
9,999 ms slot	Intervalle 9,999 ms
3,906 ms idle	Inactif 3,906 ms
11,719 ms slot	Intervalle 11,719 ms
Time	Temps

Figure 77 – Durées d'intervalle de temps et temporisation

NOTE 2 Calculs pour les temps inactifs à la Figure 77:

a) $10\,485 \times 25 \times 2^{-20}$ s = 249,982 ms. Soustraire à 250 ms pour obtenir 0,018 ms;

b) $12\,288 \times 21 \times 2^{-20}$ s = 246,094 ms. Soustraire à 250 ms pour obtenir 3,906 ms;

NOTE 3 L'intervalle de temps visant étroitement à approcher 10 ms a une durée de 10 485 unités, ou 9,999 275 ms.

La période inactive n'est pas utilisée par le système pour les opérations programmées; il s'agit simplement d'une courte période insérée pour s'assurer que les intervalles de temps s'alignent et se répètent à des intervalles de 250 ms. Cela permet au gestionnaire de système d'organiser facilement l'ensemble des intervalles de temps qui se répètent à des multiples exacts de 250 ms.

Les périodes de saut lent peuvent s'étendre sur un intervalle d'alignement. Dans de tels cas, les périodes de saut lent ne sont pas interrompues par des périodes inactives, c'est-à-dire, si une période de saut lent traverse la frontière d'un intervalle d'alignement, la radio du récepteur continue de fonctionner à travers la période inactive.

9.1.9.1.5 Temps d'intervalle de temps programmé

Chaque intervalle de temps de DL a un temps de début programmé en secondes fractionnaires, qui s'appelle le temps d'intervalle de temps programmé. Les protocoles tant de

DL que de couche supérieure utilisent ce sens coordonné de temps comme support de sécurité, assurant ainsi une protection contre le rejet. Voir 7.3.2.6.

Puisque les intervalles de 250 ms alignent avec des quarts de seconde de TAI, la temporisation d'intervalle de temps programmé peut être dérivée de la durée de l'intervalle de temps. Par exemple, si la durée d'intervalle de temps est 9,999 ms, les temps d'intervalle de temps programmé dans le deuxième quart de seconde du temps TAI t sont t + 0,250 000 s, t + 0,259 999 s, t + 0,269 998 s, etc.

NOTE Cette représentation à base multiple du temps facilite l'interconversion à des couches supérieures, où les PDU s'étendent souvent sur des sous-réseaux D avec des débits de données différents, des durées différentes d'intervalle de temps et/ou des protocoles de DL différents.

Le temps de début d'intervalle de temps programmé est en unités de 2^{-20} s (précision d'une microseconde). Les DL utilisent communément des horloges internes basées sur un cristal "de montre" très faible puissance et très précis de 2^{15} Hz (32 kiHz). Des mises en œuvre peuvent, et généralement le feront, arrondir le temps de début d'intervalle de temps programmé au top d'horloge de 32 kiHz le plus proche. Cet arrondi est autorisé sur une base d'intervalle de temps à intervalle de temps, sans réajuster le programme d'intervalle de temps sous-jacent basé sur des unités de 2^{-20} s Cependant, cet arrondissement à 32 kiHz n'est pas autorisé à s'accumuler au-dessus d'une période de 250 ms, conformément au Tableau 101. Cette considération est incluse au sein de la gigue de ±96 µs autorisée par la présente norme; voir 9.4.3.3.1.

Tableau 101 – Approximation de la temporisation nominale avec une horloge de 32 kHz

Décalage de l'intervalle de temps nominal (ms)	2^{-20} s (comptes d'horloge)	2^{-15} s (comptes d'horloge, arrondis)	Décalage de l'intervalle de temps réel (ms)
0	0	0	0
10	10 485 × 1	328	10,010
20	10 485 × 2	655	19,989
30	10 485 × 3	983	29,999
•••	•••	•••	•••
200	10 485 × 20	6 553	199,982
210	10 485 × 21	6 881	209,991
220	10 485 × 22	7 208	219,971
230	10 485 × 23	7 536	229,980
240	10 485 × 24	7 864	239,990

9.1.9.2 Propagation de temps DL

9.1.9.2.1 Généralités

La DLE utilise le temps TAI pour son fonctionnement interne, et fournit une notion de temps TAI comme service (par le DMAP) aux voisins sans fil conformes à la présente norme.

Dans un sous-réseau D, les DLE peuvent prendre trois fonctions dans le processus de propagation de temps:

- destinataire d'horloge de DL, un récepteur des mises à jour périodiques d'horloge par le biais de la DL; ou
- source d'horloge de DL, un fournisseur de mises à jour périodiques d'horloge aux voisins de DL; ou
- répéteur d'horloge de DL, un destinataire d'horloge de DL qui agit également comme source d'horloge de DL pour certains de ses voisins.

Des informations complémentaires relatives à la propagation de temps peuvent être trouvées en 6.3.10.

9.1.9.2.2 Stabilité d'horloge de DLE

La stabilité d'horloge de la DLE est la stabilité nominale d'horloge d'un appareil sans fil, en présence des corrections périodiques de temps issues du sous-réseau D.

La DLE, une fois configurée comme destinataire d'horloge, s'appuie sur le sous-réseau D pour fournir en mode sans fil les mises à jour périodiques de temps. Elle peut également utiliser ces mises à jour de temps dans l'étalonnage de son horloge interne pour rendre compte de conditions telles que la température, le vieillissement, et la tension électrique.

La référence temporelle pour un sous-réseau D provient d'un ou plusieurs maîtres d'horloge, qui fournissent le temps à croissance monotone à un taux qui suit étroitement le temps réel.

Lorsque la DLE rejoint le sous-réseau D, la DLE rapporte sa stabilité d'horloge au gestionnaire de système comme étant dlmo.DeviceCapability.ClockStability (appelée ClockStability ici), qui est dans les unités de parties par million (1×10^{-6}). ClockStability s'applique à n'importe quelle période arbitraire de 30 s au cours de la vie prévue de l'appareil, dans toutes les conditions qu'un utilisateur pourrait raisonnablement attendre de spécifications publiées de l'appareil.

Par exemple, si ClockStability rapporte qu'une DLE a une horloge d'une instabilité maximale de 10×10^{-6} s/s, alors l'horloge de la DLE doit être stable à $\pm$ 300 μ ms près au cours de n'importe quelle période de 30 s arbitrairement sélectionnée.

ClockStability est rapportée comme une enveloppe. Par exemple, si ClockStability rapporte qu'une DLE a une horloge d'une instabilité maximale de 10×10^{-6} s/s, alors l'horloge de la DLE doit être stable à $\pm$ 300 μ ms près à tout instant, au cours de n'importe quelle période de 30 s arbitrairement sélectionnée. L'intention en est de permettre la prise en charge d'appareils qui sont soumis à des chocs environnementaux occasionnels susceptibles de causer de petites discontinuités d'horloge. Les petites discontinuités sont acceptables, tant qu'elles ne s'accumulent pas pour dépasser $\pm$ClockStability $\times$ 30 μs à un moment quelconque sur une période de 30 s quelconque.

NOTE 1 ClockStability, spécifiée en unités de 1×10^{-6}s/s, est l'équivalent de 1 μs/s.

NOTE 2 Les DLE voisines peuvent avoir des horloges qui dérivent dans des sens opposés. Par conséquent, dans le cas le plus défavorable, ClockStability est additive entre des paires de DLE voisines. Dans la pratique, les répétiteurs d'horloge sont corrigés périodiquement, ce qui diminue cet effet.

La norme suppose que la dérive d'horloge est négligeable au sein d'un intervalle de temps.

ClockStability est rapporté au gestionnaire de système sans mises en garde. Si l'appareil est spécifié pour fonctionner sous contrainte environnementale, telle que des valeurs extrêmes de température ou de choc mécanique, alors ClockStability doit refléter les performances sous une telle contrainte.

L'attribut dlmo.ClockExpire, configuré par le gestionnaire de système, fournit le nombre maximal de secondes pendant lesquelles la DLE peut fonctionner en toute sécurité en l'absence d'une mise à jour d'horloge. Normalement, le gestionnaire de système s'arrange pour qu'une DLE maintienne la synchronisation d'horloge comme un sous-produit de la communication normale. Cependant, lorsque la DLE échoue à recevoir une mise à jour d'horloge pendant une période de temps prolongée, définie par ClockExpire, il convient que la DLE interroge activement une source d'horloge de DL pour une mise à jour du temps. Le manquement à le faire conduira finalement à la perte de la synchronisation du temps avec le sous-réseau D. ClockExpire prend par défaut une valeur qui est appropriée pour être utilisée au cours du processus de rattachement.

Il convient qu'un répétiteur d'horloge n'envoie pas de corrections d'horloge à ses voisins par des DPDU ACK/NAK s'il n'a pas lui-même reçu une correction d'horloge pendant une période qui excède l'attribut ClockExpire. Si l'horloge d'un répétiteur d'horloge a expiré et il est interrogé pour une mise à jour de temps, il convient qu'il réponde par un NAK1.

Il convient qu'une DLE avec une horloge expirée ne soit pas utilisée comme un répétiteur d'horloge, mais elle peut continuer à fonctionner dans le sous-réseau D, bien que ce soit avec un risque potentiel de perdre la synchronisation avec ses voisins. L'attribut dlmo.ClockTimeout fournit la quantité maximale de temps pendant laquelle une DLE peut raisonnablement continuer de fonctionner dans un sous-réseau D en l'absence d'une mise à jour d'horloge. Si la DLE n'a pas reçu une mise à jour d'horloge pendant une période de temps qui excède DLTimeout, la DLE peut raisonnablement se réinitialiser d'elle-même à l'état "provisioned "et initier une recherche d'un nouveau sous-réseau D.

NOTE 3 Une DLE fonctionnant dans une configuration de saut de voie lent est capable d'être configurée pour retenir une connexion de sous-réseau D pendant des périodes prolongées de temps, même avec une horloge qui a dérivé à travers les intervalles de temps.

9.1.9.2.3 Sources d'horloge préférentielles et secondaires

Le gestionnaire de système configure chaque destinataire d'horloge de DL pour traiter un ou plusieurs de ses voisins comme des sources d'horloge de DL. De telles sources d'horloge de DL peuvent être désignées comme étant préférentielles ou secondaires, selon l'attribut dlmo.Neighbor[ ].ClockSource. Plusieurs voisins peuvent être désignés comme étant des sources préférentielles d'horloge de DL. Il convient qu'un destinataire d'horloge de DL réajuste sa valeur d'horloge chaque fois qu'il a une interaction avec une source préférentielle d'horloge de DL.

L'attribut dlmo.ClockStale définit une période de temps qui doit s'écouler avant qu'une DLE ne commence à accepter des mises à jour d'horloge issues de sources secondaires d'horloge de DL. Si, après une période de ClockStale, aucune mise à jour d'horloge n'est reçue en provenance d'une quelconque source préférentielle, il convient que le destinataire d'horloge de DL accepte des mises à jour d'horloge dans ses interactions avec les voisins qui sont désignés comme étant des sources secondaires d'horloge de DL. Une fois qu'un destinataire d'horloge de DL accepte une mise à jour d'horloge issue d'une source secondaire d'horloge de DL, il convient qu'il continue d'utiliser cette source secondaire d'horloge de DL jusqu'à ce que soit (a) il reçoive une mise à jour d'horloge issue d'une source préférentielle d'horloge de DL, soit (b) la source secondaire d'horloge de DL temporise.

L'attribut dlmo.ClockStale détermine l'intervalle de temporisation. Par exemple, si ClockStale est mis à 45 s par le gestionnaire de système, il convient qu'une source d'horloge de DL n'accepte pas de mises à jour d'horloge issues d'une source secondaire d'horloge de DL jusqu'à ce qu'elle n'ait pas reçu une mise à jour d'horloge issue d'une quelconque source préférentielle d'horloge de DL pendant au moins 45 s.

Chaque destinataire d'horloge de DL peut être configuré pour conserver et rapporter périodiquement des statistiques (voir 9.4.3.9) pour n'importe laquelle de ses sources préférentielles d'horloge de DL, y compris:

- Un compte des événements de temporisation d'horloge.
- Une moyenne glissante des corrections d'horloge, un nombre entier signé en unités de 2^{-20} s, indiquant un biais si différent de zéro.
- L'écart-type des corrections d'horloge, estimé en unités de 2^{-20} s, par exemple, une valeur qui rend grossièrement compte de 68 % approximativement des corrections d'horloge.
- Un compte des corrections d'horloge au-dessus de trois fois de tels écarts-types.

9.1.9.2.4 Sens partagé de temps pendant le fonctionnement du sous-réseau D

Les transactions de sous-réseau D se produisent nominalement dans un intervalle de temps de DL, sur un programme connu à la fois de l'expéditeur et du récepteur. Ce sens partagé de temps est utilisé comme support de sécurité, tant pour la compression d'en-tête que pour la protection contre le rejet.

Des sources d'horloge de DL peuvent être configurées pour émettre périodiquement des annonces de DL intégrées dans un sous-en-tête DAUX d'une DPDU Data. Chaque annonce de ce type fournit une référence de temps TAI pour le sous-réseau D. Toutes les DLE compatibles avec la norme qui reçoivent une annonce sont supposées être capables de participer à une communication synchronisée en temps avec la DLE d'annonce. Les destinataires d'horloge de DL avec des horloges relativement imprécises peuvent avoir une capacité limitée de ne communiquer qu'avec les routeurs qui utilisent le saut de voie lent.

L'authentification de couche MAC exige des clés de sécurité partagées, le sens partagé du temps, et la connaissance de l'adresse EUI64Address d'un voisin. Ces informations sont acquises par étapes au cours du processus de rattachement, avec des clés de sécurité fournies à la fin. La DL utilise le chiffrement normalisé par blocs (habituellement AES-128), ainsi qu'une clé de sécurité bien connue, au cours du processus de rattachement afin de fournir une vérification d'intégrité améliorée (mais pas la sécurité). Cela est décrit en plus de détails en 7.4.

Dans le saut de voie discrétisé, les initiateurs de transaction et les destinataires de transaction partagent un sens intrinsèque de l'intervalle de temps qui est utilisé; autrement, les DLE ne seraient pas capables de communiquer. Chaque intervalle de temps a un temps de début programmé qui est connu par toutes les DLE participantes.

Dans le saut de voie lent, une DLE avec un sens inexact du temps de DL émettra nominalement dans un intervalle de temps particulier, en fonction de son propre sens du temps. Cependant, une telle DLE est autorisée à manquer sa cible, et peut réellement émettre dans n'importe quel intervalle de temps dans les limites de la période de saut de voie lent. Par conséquent, les DPDU Data en monodiffusion qui sont émises en utilisant une supertrame de saut de voie lent doivent inclure un octet supplémentaire dans l'en-tête de la DPDU Data pour indiquer quel intervalle de temps au sein de la période de saut de voie lent était prévu. Cela permet à la DLE réceptrice d'accorder son sens d'intervalle de temps avec celui de la DLE émettrice, appliquant la correction de temps à travers les intervalles de temps pour valider l'exactitude de l'horloge de l'initiateur de la transaction, et d'utiliser le temps de début programmé de l'intervalle de temps comme support de sécurité.

Pour les besoins de la sécurité, un temps de début programmé (qui n'est pas le temps de début d'une DPDU Data obtenue) d'un intervalle de temps est transmis au DSC pour être utilisé dans des opérations de sécurité relatives à la DL. Voir 9.1.11. Dans la plupart des cas, l'intervalle de temps des DPDU Data et l'intervalle de temps de n'importe quelle DPDU ACK/NAK répondeuse sont les mêmes. Il y a une exception, qui se produit pour le saut de voie lent lorsque l'horloge d'une des DLE a dérivé pour entrer dans un intervalle de temps différent. Dans ce cas, le décalage de saut de voie de la DLE d'acquittement doit être inclus dans le DMXHR de la DPDU ACK/NAK (Tableau 117) pour identifier sans ambiguïté le temps exact qui doit être utilisé pour des opérations de sécurité, même si la DLE d'acquittement n'agit pas comme source d'horloge de DL pour le destinataire de la DPDU ACK/NAK DPDU. L'absence (à savoir la non-inclusion) champ décalage de saut de voie dans la DPDU ACK/NAK implique que la DLE qui est la source de la DPDU Data et la DLE qui est la source de la DPDU ACK/NAK utilisent le même intervalle de temps.

9.1.9.3 Synchronisation du temps par paires

9.1.9.3.1 Généralités

Des mises à jour d'horloge sont propagées à travers le sous-réseau D au cours des communications normales dans les limites d'un intervalle de temps ou d'une période de saut

de voie lent. Ces blocs modules de base sont soumis à un effet de levier par le gestionnaire de système pour arranger la propagation du temps de sous-réseau D.

Les sources d'horloge de DL utilisent trois mécanismes généraux pour propager des mises à jour d'horloge vers leurs voisins.

- Une source d'horloge de DL initie une DPDU Data qui inclut une annonce. Le temps est acheminé en fonction de l'instant où la DPDU Data est émise.
- Une source d'horloge de DL acquitte une DPDU Data reçue dans un intervalle de temps. Le temps est acheminé en mesurant l'instant où la source d'horloge de DL détecte le début de la DPDU Data et en faisant écho au résultat de cette mesure dans la/les DPDU ACK/NAK.
- Une source d'horloge de DL acquitte une DPDU Data dans les limites d'une période de saut de voie lent. Le processus est similaire à l'acquittement au sein d'un intervalle de temps, avec l'addition d'un octet pour identifier de façon univoque l'intervalle de temps si besoin est.

La norme s'appuie sur des horloges stables, en particulier dans des routeurs de terrain, pour le fonctionnement fiable du sous-réseau D. Pour des exigences relatives à la stabilité d'horloge de DL, voir le Tableau B.8.

Un destinataire d'horloge de DL maintient une liste de sources valides d'horloge de DL voisines. S'il reçoit une mise à jour d'horloge issue d'une source désignée d'horloge de DL, il utilise les données pour mettre à jour son horloge.

Lorsqu'une DLE découvre un sous-réseau D, elle acquiert le temps du sous-réseau D et utilise cette référence pour la communication ultérieure. La DLE doit alors resynchroniser périodiquement son horloge interne à l'horloge de sous-réseau D. Ces opérations de resynchronisation sont incrémentielles, en fonction des décalages dans un intervalle de temps avec une référence temporelle programmée connue à la fois de la source d'horloge de DL et du destinataire d'horloge de DL.

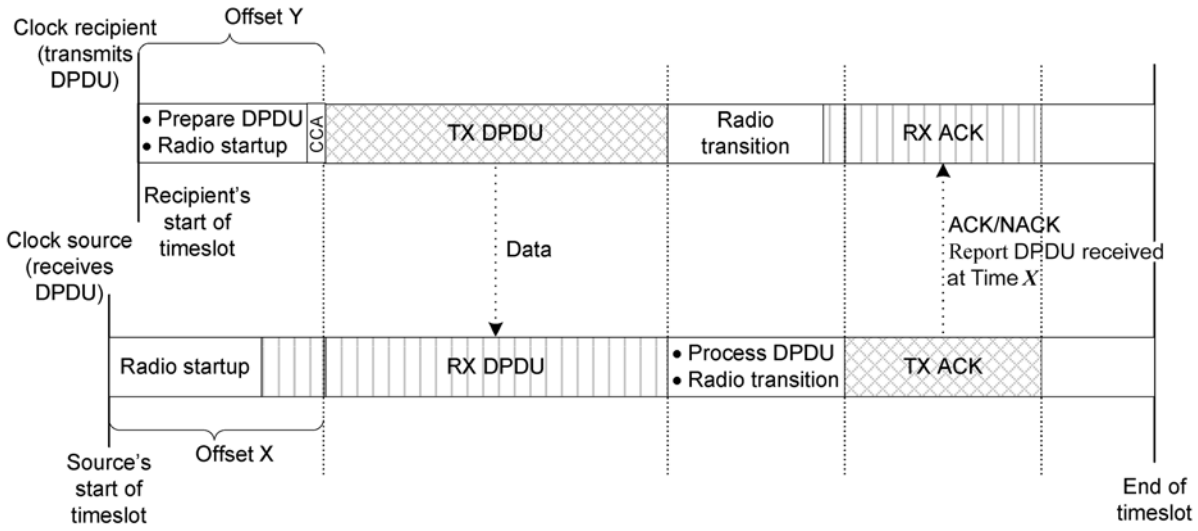
Chaque fois qu'une DLE interagit avec l'une de ses sources désignées d'horloge de DL, elle reçoit des informations mises à jour relatives aux horloges. Des réajustements d'horloge sont inclus dans les DPDU ACK/NAK, acheminant ainsi les informations relatives au temps vers le destinataire. Des mises à jour d'horloge sont également incluses dans le sous-en-tête DAUX des DPDU Data d'annonce provenant de la source d'horloge de DL.

Ainsi, une DLE peut recevoir des mises à jour d'horloge comme un sous-produit des données de routage par l'intermédiaire d'une source d'horloge de DL. En variante, si une DLE a besoin de plus fréquentes mises à jour d'horloge, elle peut être configurée pour recevoir des mises à jour d'horloge en permettant à son récepteur de fonctionner à des temps coïncidant avec des DPDU Data programmées périodiques provenant de sources d'horloge de DL qui incluent le sous-en-tête DAUX avec une annonce.

Une DLE peut acquérir une mise à jour d'horloge en envoyant une DPDU Data avec une DSDU de longueur zéro vers une source d'horloge de DL. Une source d'horloge de DL doit acquitter une telle DPDU Data et ensuite la rejeter.

9.1.9.3.2 La source d'horloge acquitte la réception d'une DPDU Data au sein d'un intervalle de temps

Lorsqu'une DLE émet une DPDU Data en monodiffusion (voir Figure 81) vers une source d'horloge de DL, elle peut recevoir une mise à jour d'horloge dans la DPDU ACK/NAK, comme montré à la Figure 78.



Légende

Anglais	Français
Clock recipient (transmits DPDU)	Récepteur d'horloge (émet la DPDU)
Offset Y	Décalage Y
Prepare DPDU	Préparer la DPDU
Radio startup	Démarrage radio
Radio transition	Transition radio
Clock source (receives DPDU)	Source d'horloge (reçoit la DPDU)
Recipient's start of timeslot	Début d'intervalle de temps récepteur
Data	Données
ACK/NACK Report DPU received at time X	ACK/NACK Rapporter DPDU reçue au temps X
Process DPDU	Traiter DPDU
Source's start of timeslot	Début d'intervalle de temps source
Offset X	Décalage X
TX ACK	ACK émission
RX DPDU	DPDU réception
TX DPDU	DPDU émission
RX ACK	ACK réception
End of timeslot	Fin d'intervalle de temps

Figure 78 – La source d'horloge acquitte la réception d'une DPDU Data

La DPDU ACK/NAK issue de la source de données de DL inclut le temps de début de l'émission d'origine rapporté comme étant un décalage de temps (avec une précision de microseconde, en unités de 2^{-20} s) du temps de début de la DPDU Data (à savoir le temps de début de la PhSDU) par rapport au temps de début d'intervalle de temps programmé tel que mesuré par la source d'horloge de DL.

Une transaction D en monodiffusion donnée se produit sur une seule voie, avec la DPDU Data et la/les DPDU ACK/NAK DPDU toutes émises sur la même voie.

Ces informations de synchronisation d'horloge sont ciblées à la DLE qui est à l'origine de la DPDU Data. Alors que d'autres DLE peuvent entendre par hasard et mettre à jour leurs horloges en fonction des mêmes informations, la conception n'est pas optimisée pour cette utilisation.

Alors que les détails internes d'une DLE relatifs à la synchronisation d'horloge ne sont pas spécifiés par la présente norme, la transaction est destinée à prendre en charge un processus de synchronisation d'horloge qui fonctionne généralement comme suit:

a) Le destinataire d'horloge de DL envoie une DPDU Data à la source d'horloge de DL et enregistre qu'elle a envoyé la DPDU Data à un décalage de temps Y.

b) La source d'horloge de DL reçoit la DPDU Data virtuellement au même instant et enregistre qu'elle a reçu la DPDU Data à un décalage de temps X.

c) Le diagramme montre le cas où l'horloge du destinataire d'horloge de DL tourne plus vite que la source d'horloge de DL. Dans ce cas, $X > Y$. Si la source d'horloge de DL tourne plus vite, $X < Y$. Le temps de la source d'horloge de DL est supposé être correct.

d) Dans la DPDU ACK/NAK, la source d'horloge de DL rapporte qu'elle a reçu la DPDU Data à un décalage de temps X.

e) Le destinataire d'horloge de DL applique une correction de temps qui est calculée comme étant $(Y - X)$.

Le résultat montré dans la Figure 78 est que les intervalles de temps commencent à des temps différents, mais finissent au même temps. Une mise en œuvre réelle peut retarder l'application des corrections de temps, par exemple en réajustant progressivement l'horloge. Voir 9.1.9.2.2.

9.1.9.3.3 Une source d'horloge initie une DPDU Data qui inclut une annonce

Une source d'horloge de DL est souvent la source d'une DPDU Data. Tel est généralement le cas lorsqu'une source d'horloge de DL émet une annonce prévue qui inclut le temps TAI. La charge utile de la DPDU Data peut être sans rapport avec la synchronisation; les informations relatives au temps TAI sont contenues dans le sous-en-tête DAUX et elles peuvent donc être entendues par hasard par n'importe lequel des voisins de la source d'horloge de DL.

Plusieurs DLE peuvent être configurées pour activer leurs destinataires simultanément par anticipation d'une DPDU Data programmée acheminant une annonce.

La présente norme exige qu'une source d'horloge de DL soit capable de commander précisément le moment où des DPDU Data d'annonce sont émises, avec une précision des $\pm$ 96 μs (3 octets), en étant référencées à l'horloge interne de la DLE. Voir 9.4.3.3.1.

9.1.9.3.4 Une source d'horloge acquitte une DPDU Data dans les limites d'une période de saut de voie lent

Comme noté précédemment, les sources d'horloge de DL utilisent des DPDU ACK/NAK pour rapporter le temps avec une résolution de microseconde (2^{-20} s), comme étant un décalage par rapport au temps de début nominal programmé d'un intervalle de temps. L'identification de l'intervalle de temps est supposée être non ambiguë, connue à la fois de l'expéditeur et des destinataires. Ainsi, seul le décalage est communiqué dans les mises à jour d'horloge.

Cependant, au cours des périodes de saut de voie lent, l'horloge interne d'un destinataire d'horloge de DL peut avoir dérivé des dizaines ou des centaines de millisecondes, ou plus, par rapport à l'horloge de l'expéditeur, si bien que la présomption de l'identification de l'intervalle de temps pourrait être incorrecte. Par conséquent, un octet supplémentaire, identifiant un intervalle de temps spécifique, est ajouté aux en-têtes de DPDU dans les limites des périodes de saut de voie lent, tel que spécifié dans le DHDR de la DPDU. L'intervalle de temps initial au sein d'une période de saut de voie lent est défini comme ayant un décalage de zéro.

Au sein des périodes de saut de voie lent, il convient que les DLE avec un sens exact du temps fonctionnent à l'intérieur des frontières d'intervalle de temps. Cependant, les DLE sans sens exact du temps pourraient ne pas être capables de respecter les frontières d'intervalle de temps, et pourraient même ne pas savoir quel intervalle de temps elles utilisent

réellement. Un décalage d'intervalle de temps dans chaque en-tête de DPDU permet au récepteur de reconstruire le sens du temps de l'expéditeur et vice-versa (voir 9.3.3.3).

9.1.9.3.5 Audit de la qualité d'une horloge de voisin

Lorsqu'une DLE rejoint le sous-réseau D, elle rapporte ses caractéristiques de stabilité d'horloge à la fonction de gestion du système. Dans leurs interactions normales avec ces DLE, les sources d'horloge de DL et les destinataires d'horloge de DL peuvent être configurés par le système pour recueillir le diagnostic, sur une base voisin par voisin, pour auditer ces spécifications nominales (voir 9.4.3.9).

9.1.9.3.6 Réajustements d'horloges discontinus

Sous certaines conditions, il peut être nécessaire pour le gestionnaire de système de réajuster l'ensemble de toutes ses horloges de DLE par dizaines ou centaines de millisecondes, ou plus, par exemple lorsque deux sous-réseaux D sont rejoints. Le gestionnaire de système peut programmer un réajustement d'horloge discontinu qui se produit à un temps TAI particulier, en positionnant l'attribut dlmo.TaiAdjust sur chacune de ses DLE. Au temps désigné, toutes les DLE doivent réajuster leurs horloges en les avançant ou retardant d'une quantité de temps spécifiée.

Le réajustement d'horloge a un certain nombre d'impacts sur le système qu'il convient de prendre en compte chaque fois que cette caractéristique est utilisée.

- Le modèle de sécurité dans la présente norme n'autorise pas que le temps recule. Le temps est utilisé dans le nonce de sécurité, qui ne peut jamais être répété dans une quelconque couche de communication normalisée. En conséquence, il doit y avoir une interruption de service, égale à l'amplitude du réajustement plus au moins un intervalle de temps, si le temps est réajusté à un temps antérieur.
- Les rapports échelonnés décaleront leur temps de la quantité du réajustement d'horloge, à moins que des contrats correspondants ne soient révisés en tandem avec le réajustement d'horloge.
- Les DLE qui ne reçoivent pas la correction de temps avant l'heure de basculement peuvent perdre la synchronisation avec le fonctionnement du sous-réseau D, et peuvent avoir besoin de redécouvrir leurs voisins.

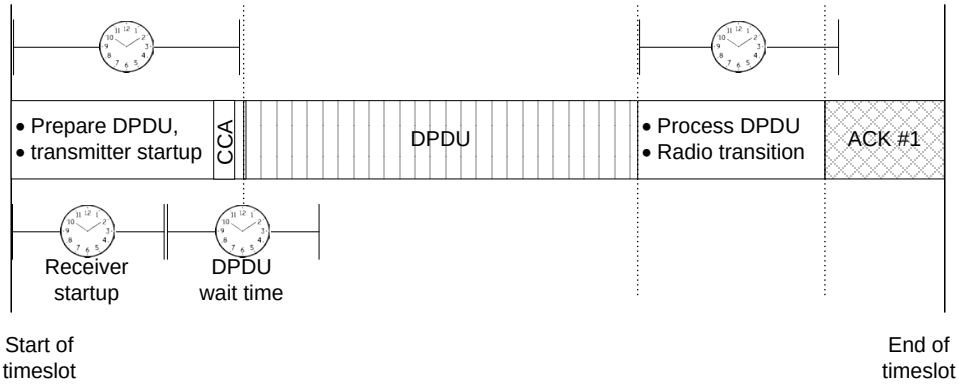
9.1.9.4 Transactions au sein des modèles d'intervalle de temps

9.1.9.4.1 Généralités

NOTE Les transactions sont également décrites en 4.6.11.

La temporisation de transaction au sein d'un intervalle de temps de DL est spécifiée par des modèles d'intervalle de temps. La norme définit les modèles d'intervalle de temps par défaut qui sont nécessaires à la DLE pour ses interactions avec une DLE de configuration sans fil. Des modèles complémentaires d'intervalle de temps peuvent être ajoutés par le gestionnaire de système pendant ou après le processus de rattachement. (Voir 9.4.3.3 pour plus d'informations relatives aux modèles d'intervalle de temps.)

La Figure 79 montre certains aspects traités par les modèles d'intervalle de temps de DL.



Légende

Anglais	Français
Prepare DPDU	Préparer DPDU
Transmitter startup	Démarrage émetteur
Process DPDU	Traiter DPDU
Radio transition	Transition radio
ACK#1	ACK n°1
Receiver startup	Début récepteur
DPDU wait time	Temps d'attente DPDU
Start of timeslot	Début d'intervalle de temps
End of timeslot	Fin d'intervalle de temps

Figure 79 – Attributs de temporisation de transaction

Comme montré à la Figure 79, les modèles d'intervalle de temps définissent la temporisation pour les opérations telle que le temps d'attente de réception des DPDU Data, et le temps d'inversion (traitement de réception de DPDU Data et transition radio) entre la réception d'une DPDU Data et l'émission d'une ACK/NAK DPDU.

Généralement, il y a deux types de modèles de transaction: émission et réception. Un modèle d'initiateur de transaction spécifie une plage de temps pour commencer l'émission d'une DPDU Data, pour vérifier l'activité avant émission, et le placement relatif de toutes les éventuelles DPDU ACK/NAK DPDU dans l'intervalle de temps. Un modèle de récepteur de transaction spécifie l'intervalle pendant lequel une DPDU Data reçue peut commencer à arriver et, ainsi, quand temporiser si aucun début de DPDU Data n'est détecté. Il spécifie également la temporisation de toutes les éventuelles DPDU ACK/NAK envoyées par des répondeurs de transaction.

Les modèles d'intervalle de temps par défaut couvrent les transactions de ligne de base nécessaires pour interagir avec une DLE de configuration, qui sont utilisables au cours du processus de rattachement. Des modèles par défaut peuvent également être utilisés par des DLE rattachées pour des transactions générales. Des modèles complémentaires peuvent être configurés dans la DLE pour prendre en charge des caractéristiques telles que:

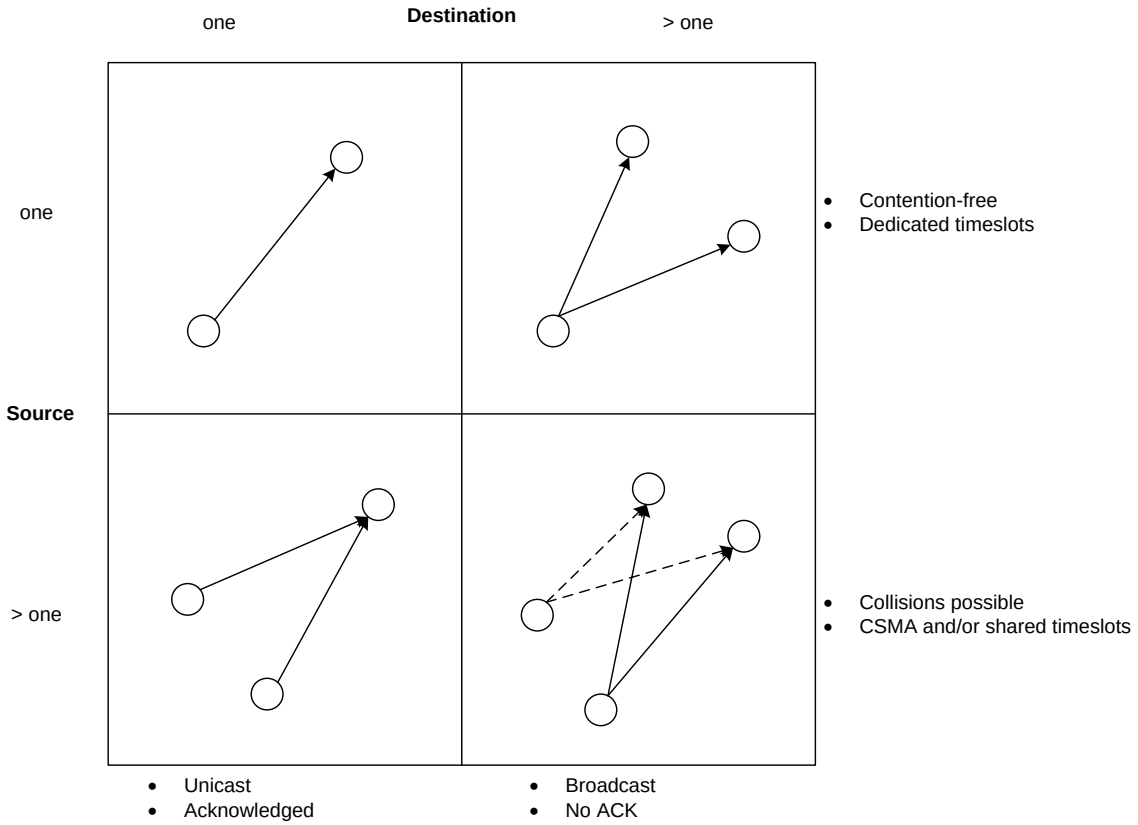
- périodes d'accès multiple avec détection de porteuse avec évitement de collision (CSMA/CA) au début d'un intervalle de temps (voir 9.1.9.4.8); et
- temporisation prolongée de DPDU ACK/NAK par rapport à la fin de la DPDU Data ou par rapport à la fin de l'intervalle de temps (voir 9.1.9.4.6).

Par convention dans la présente norme, la temporisation de modèle d'intervalle de temps est spécifiée en fonction des temps de début et de fin de DPDU Data et ACK/NAK et de la fin de l'intervalle de temps. La temporisation de PhPDU, dépendante des détails de la couche

physique qui achemine chaque DPDU, peut être inférée à partir des ces temps de début et de fin de DPDU (voir 9.4.3.3).

9.1.9.4.2 Vue d'ensemble d'une transaction D

Comme montré à la Figure 80, la DL prend en charge les transactions en modes tant monodiffusion que diffusion.



Légende

Anglais	Français
One	Une seule
Destination	Destination
>one	>une
Contention-free	Sans conflits
Dedicated timeslots	Intervalles de temps dédiés
Source	Source
Collisions possible	Collisions possibles
CSMA and/or shared timeslots	CSMA et/ou intervalles de temps partagés
Unicast	Monodiffusion
Acknowledged	Acquitté
Broadcast	Diffusion
No ACK	Pas d'ACK

Figure 80 – Intervalles de temps de transaction dédiés et partagés

Les transactions en monodiffusion, indiquées dans la moitié gauche de la Figure 80, peuvent utiliser des intervalles de temps dédiés, par exemple, lorsqu'une DLE rend répétitivement compte d'un programme. La duodiffusion est une variante de la monodiffusion, dans laquelle

un second destinataire est programmé pour entendre par hasard la Data DPDU et fournit une seconde DPDU ACK/NAK. La duodiffusion est montrée graphiquement à la Figure 84.

La réception d'une DPDU ACK (acquittement positif) par l'initiateur de la transaction indique que le destinataire de la transaction a reçu avec succès la DPDU Data et qu'il convient que l'initiateur de la transaction marque la transaction comme étant achevée. Les transactions en monodiffusion et duodiffusion exigent l'émission de DPDU ACK/NAK en réponse à une telle réception.

Les transactions en diffusion, indiquées dans la moitié droite de la Figure 80, peuvent également utiliser des intervalles de temps dédiés, comme pour les annonces programmées. Les transactions en diffusion ne peuvent pas utiliser un acquittement immédiat de DL (à savoir par l'intermédiaire d'une DPDU ACK/NAK).

Comme montré dans le quadrant inférieur gauche de la Figure 80, les transactions en monodiffusion et en duodiffusion peuvent utiliser des intervalles de temps partagés, comme au sein de périodes de saut lent. Les intervalles de temps partagés sont communément utilisés pour les répétitions de tentative, les demandes de rattachement, les rapports relatifs à des exceptions et le trafic en salves.

Comme montré dans le quadrant inférieur droit de la Figure 80, les transactions en diffusion telles que des sollicitations peuvent également utiliser des intervalles de temps partagés.

Le terme sans conflits à la Figure 80 n'est pertinent qu'au sein de l'allocation d'intervalles de temps de sous-réseau D. Si deux sous-réseaux D, ou des DLE au sein du même sous-réseau D, allouent des intervalles de temps d'une manière non coordonnée, l'accès n'est pas sans conflits. De même, d'autres utilisateurs du spectre de 2,4 GHz, tels que WiFi, Bluetooth®[9], ZigBee, IEC 62591 et IEC 62601, peuvent potentiellement interférer aussi. La coexistence améliorée avec ces systèmes non coordonnés est réalisée en utilisant la CCA (voir 9.1.9.4.3) pour vérifier la voie avant émission.

L'utilisation de la diffusion dans la présente norme est limitée à ces opérations de DL: annonces et sollicitations. Les annonces et les sollicitations, utilisées pour la découverte de sous-réseau D, sont décrites en 9.1.13.

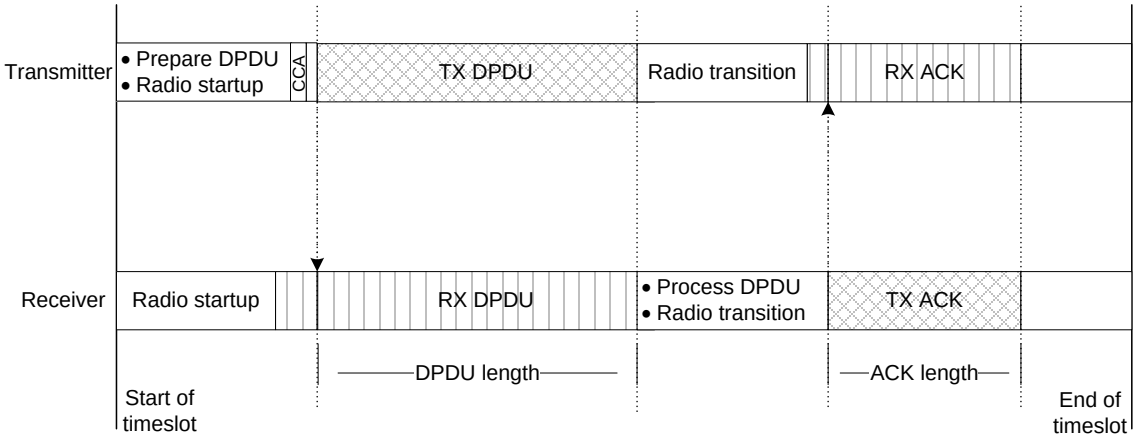
NOTE La diffusion telle que décrite ici n'utilise pas les MPDU en diffusion définies dans l'IEEE 802.15.4. Voir 9.1.5. La diffusion et la multidiffusion, comme services d'AL, ne sont pas prises en charge dans la présente version de la norme.

9.1.9.4.3 Transaction en monodiffusion

La Figure 81 montre une transaction en monodiffusion.

---

9 Bluetooth® est l'appellation commerciale d'un produit distribué par Bluetooth Special Interest Group. Ces informations sont données à titre indicatif aux utilisateurs de la présente norme et ne constituent pas une approbation de ces produits par l'IEC. Des produits équivalents peuvent être utilisés s'il est démontré qu'ils permettent d'obtenir les mêmes résultats.



Légende

Anglais	Français
Transmitter	Emetteur
Prepare DPDU	Préparer DPDU
Radio startup	Démarrage radio
TX DPDU	DPDU émission
Radio transition	Transition radio
RX ACK	ACK réception
Receiver	Récepteur
RX DPDU	DPDU réception
Process DPDU	Traiter DPDU
TX ACK	ACK émission
DPDU length	Longueur DPDU
ACK length	Longueur ACK
Start of timeslot	Début d'intervalle de temps
End of timeslot	Fin d'intervalle de temps

Figure 81 – Transaction en monodiffusion

Avant qu'une DLE n'émette une DPDU Data dans un intervalle de temps, elle prépare la DPDU Data pour l'émission, généralement avec la sécurité de DL. Avant l'émission, la DLE est normalement configurée pour accomplir une évaluation de voie libre (CCA) de l'espace radio.

La CCA doit être mise en œuvre conformément à la description de l'IEEE 802.15.4. La présente norme spécifie une période de détection de 8 symboles. La CCA doit être accomplie telle que configurée dans le champ de modèle d'intervalle de temps dlmo.TsTemplate[ ].CCAmode (voir Tableau 163), où les choix, énumérés par leur codage, sont:

- CCA Mode 4 (Aloha);
- CCA Mode 1 (énergie au-dessus du seuil);
- CCA Mode 2 (détection de porteuse seulement);
- CCA Mode 3 (détection de porteuse et/ou énergie au-dessus du seuil).

La conformité à l'IEEE 802.15.4 exige qu'au moins un des modes de CCA soit pris en charge. La conformité à l'ETSI EN 300 328 exige que le Mode 1 CCA soit pris en charge. Si une valeur différente de zéro pour dlmo.TsTemplate[ ].CCAmode est sélectionnée et le mode sélectionné n'est pas pris en charge par la DLE, la DLE peut choisir un mode CCA différent

pris en charge par la DLE et autorisé par le régime de réglementation applicable. Les modes CCA pris en charge par la DLE sont spécifiés dans dlmo.DeviceCapability.SupportedCCAmodes; voir 9.4.2.23.

L'IEEE 802.15.4 permet au Mode 3 CCA d'être mis en œuvre soit comme AND (c'est-à-dire ET), soit comme OR (c'est-à-dire OU), du Mode 1 CCA et du Mode 2 CCA. Lorsque le Mode 3 CCA est mis en œuvre comme AND du Mode 1 CCA et du Mode 2 CCA, il peut être utilisé en lieu et place du Mode 1 CCA dans les régimes de réglementation qui exigent le Mode 1 CCA. Lorsque le Mode 3 CCA est mis en œuvre comme OR du Mode 1 CCA et du Mode 2 CCA, il ne peut pas être utilisé en lieu et place du Mode 1 CCA dans les régimes de réglementation qui exigent le Mode 1 CCA.

Si un mode CCA spécifique est exigé par la réglementation, mais n'est pas pris en charge par la mise en œuvre, la DLE ne doit pas initier de transactions.

Si la CCA rapporte un support occupé, la transaction d'émission doit être abandonnée.

L'utilisation de la CCA telle que définie par l'IEEE n'est pas censée exclure des vérifications CCA complémentaires qui pourraient être prises en charge par des appareils évolués. Par exemple, si un routeur dorsal est capable de détecter la modulation IEEE 802.11:2012, la DLE peut raisonnablement exercer un effet de levier sur cette capacité pour détecter et rapporter un support occupé.

Il convient que la CCA soit parachevée 192 μs avant le début de l'en-tête de couche physique.

Si la transaction D exige une DPDU ACK/NAK, la DLE émettrice active son récepteur après l'achèvement de l'émission, à un temps spécifié dans TsTemplate. Si une DPDU ACK est reçue, la DPDU Data émise est supprimée de la file d'attente d'émission de DEL.

Lorsqu'une DLE est programmée pour recevoir une DPDU Data, elle active son récepteur de PhLE au temps spécifié dans le TsTemplate et attend la PhPDU prévue. Si elle détecte un SHR et un PHR IEEE 802.15.4 valides, elle continue de tenter de recevoir la PhPDU complète. Elle traite ensuite les DPDU Data (y compris l'authentification de DMIC) contenues et détermine si la DPDU Data requiert un acquittement. Pour envoyer une DPDU ACK/NAK en acquittement, une DLE qui est une répondeuse de transaction active son émetteur de PhLE et envoie la DPDU ACK/NAK dans le même intervalle de temps de telle sorte que la DPDU ACK/NAK est transmise lors du temps spécifié par le modèle d'intervalle de temps pour la répondeuse principale (ou secondaire, etc.) de l'intervalle de temps en question, selon le rôle de la DLE comme répondeuse primaire (ou secondaire, etc.).

La fenêtre de temps pour chaque DPDU ACK/NAK attendue est définie par le modèle d'intervalle de temps. S'il y a un retard substantiel entre la fin de la DPDU Data et le début programmé de la DPDU ACK/NAK attendue, une mise en œuvre peut raisonnablement mettre hors tension son récepteur pendant le retard.

9.1.9.4.4 Acquittements négatifs

Un destinataire de transaction doit répondre à la réception des DPDU Data en monodiffusion par une DPDU NAK lorsqu'il ne peut pas accepter la DPDU Data à cet instant, mais l'a reçue avec succès sans autres erreurs. Les informations relatives à la synchronisation du temps peuvent être incluses dans des DPDU NAK. De même, une DPDU NAK assure que les statistiques RF journalisent correctement une émission propre. Une DPDU NAK peut être utilisée pour exercer une contre-pression comme un simple mécanisme de contrôle de flux (régulation de débit).

La DL prend en charge deux types de DPDU NAK: NAK0 et NAK1. Une DPDU NAK0 vise à indiquer des limitations de ressources dans le routeur, alors qu'une DPDU NAK1 vise à signaler des problèmes de connectivité aval dans le sous-réseau D.

La DLE doit répondre à une DPDU Data en monodiffusion par une DPDU NAK0 lorsqu'elle reçoit correctement la DPDU Data, mais ne peut pas l'accepter en raison du manque de capacité dans sa file d'attente de messages. Voir 9.1.8.5 pour un débat relatif à la file d'attente de messages de DLE. Une DLE peut également répondre par une NAK0 lorsqu'elle est configurée au-dessus de sa capacité de transmission (ForwardRate), telle que décrite en 9.4.2.23.

La DLE peut répondre à une DPDU Data en monodiffusion par une DPDU NAK1 pour appliquer une contre-pression en cas de perte de connectivité aval. Par exemple, lorsque la DLE perd la connectivité aval à tous ses prochains voisins dans un graphe spécifique et reçoit alors une DPDU Data suivant le même graphe, la DLE peut raisonnablement générer une réponse immédiate DPDU NAK1 afin d'indiquer qu'elle n'est pas capable de transmettre des DPDU Data dirigées vers le même graphe.

Lorsqu'une DLE reçoit une PDU NAK0 ou NAK1 issue d'un voisin, elle doit se replier en n'émettant pas plus de DPDU Data vers le voisin en question pendant une période dlmo.NackBackoffDur. Ce retard de repli n'inclut pas le retard des DPDU Data sans charge utile, ce qui permet à la DLE d'interroger un voisin qui est une source d'horloge de DL pour une mise à jour de temps bien que le voisin n'accepte pas des DPDU Data à l'instant en question.

Comme décrit en 9.1.9.2.2, si l'horloge d'un répétiteur d'horloge de DL a expiré et il est interrogé pour une mise à jour de temps, il convient qu'il réponde par un NAK1.

9.1.9.4.5 Notification d'encombrement explicite

La présente norme prend en charge la notification d'encombrement explicite (ECN) telle que décrite dans l'IETF RFC 3168. L'ECN fournit un mécanisme pour un routeur pour affecter le contrôle de flux de l'AL.

Comme décrit en 12.12.4.2.3, il y a un nombre limité de demandes de source de données qui peuvent attendre simultanément la réponse issue d'un puits de données. Le contrôle de flux à la source de données fonctionne en augmentant par incréments la limite sur les demandes en attente, en fonction de la réception d'acquittements en temps utile du puits de données.

L'ECN fournit un mécanisme par lequel le contrôle de flux peut être efficace sans conduire le sous-réseau D au point de défaillance. Tout routeur le long du chemin allant de la source de données au puits de données peut établir l'ECN pour indiquer que le routeur approche de sa capacité. Lorsque le puits de données reçoit l'ECN, il en fait l'écho à la source de données, qui rend alors compte de l'ECN dans sa logique de contrôle de flux.

Un indicateur d'ECN dans l'en-tête de la DPDU Data peut être positionné par tout routeur de terrain qui est confronté à un encombrement, suivant les lignes directrices dans l'IETF RFC 3168, comme signal indiquant à une source de données qu'il convient qu'elle applique le contrôle de flux pour réduire son utilisation des ressources du réseau D. Cet indicateur d'ECN est propagé vers le puits de données, tel qu'une passerelle, et retourne finalement à la source de donnée par un acquittement de TL.

En outre, la DL fournit un type spécial d'acquittement appelé ACK/ECN. Une DLE recevant un ACK/ECN le traite comme un acquittement de DL normal. En outre, la DLE peut traiter l'ACK/ECN comme étant une indication précoce que l'acquittement du puits de données inclura une ECN.

Il convient de limiter l'utilisation de l'ACK/ECN aux DPDU Data ayant une priorité de sept (7) ou inférieure, correspondant à des flux séquentiels de meilleur effort, placés en file d'attente et temps réel.

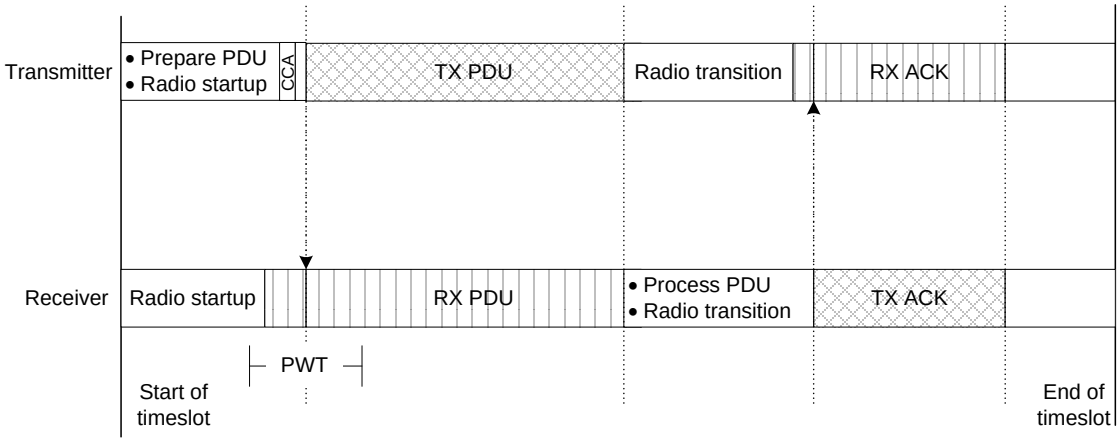
9.1.9.4.6 Temps d'attente de DPDU Data

Les temps d'horloge des DLE émettrices et réceptrices sont rarement en parfaite synchronisation. Une DLE émettrice est donc peu susceptible d'émettre une PDU (unité de données de protocole) au temps exact où une DLE réceptrice l'attend. Si une PDU est émise trop tôt, le récepteur pourrait ne pas avoir été encore activé. Si une PDU est émise trop tard, la DLE réceptrice peut avoir désactivé sa radio pour économiser de l'énergie. Le temps d'attente de PDU (PWT) est la période de temps où le récepteur est censé écouter pour détecter des PDU entrantes. Un degré particulier d'exactitude de temporisation entre les DLE est implicite dans la sélection du PWT par le gestionnaire de système.

Les DLE conformes avec la présente norme prennent en charge des PWT configurables et des durées d'intervalle de temps configurables. Le PWT n'est pas directement spécifié dans la norme, mais peut être inféré dans des modèles d'intervalle de temps à partir des temps au plus tôt et au plus tard auxquels la réception de PDU commence. Voir Tableau 161.

NOTE 1 Le présent tutoriel ne fait pas de distinction entre les temporisations de PhPDU et de DPDU. Comme spécifié en 9.4.3.3, la DL suit une convention consistant à spécifier la temporisation d'intervalle de temps en référence à la DPDU (PhSDU), pas à la PhPDU. Dans les mises en œuvre basées sur l'IEEE 802.15.4 (2,4 GHz), la détection d'en-tête de PhPDU implique la réception d'un préambule, d'un délimiteur de début de trame (SFD), et d'une longueur de trame, pour un en-tête total de PhPDU de 192 μs (6 octets) avant que la DPDU ne commence. De même, l'émission radio d'une PhPDU commence 192 μs avant le temps de début nominal de DPDU.

Des temps d'attente de PDU dans une PDU en monodiffusion sont montrés à la Figure 82. Les mêmes principes s'appliquent à d'autres types de PDU.



Légende

Anglais	Français
Transmitter	Emetteur
Prepare PDU	Préparer PDU
Radio startup	Démarrage radio
TX PDU	DPDU émission
Radio transition	Transition radio
RX ACK	ACK réception
Receiver	Récepteur
RX PDU	DPDU réception
Process PDU	Traiter DPDU
TX ACK	ACK émission
Start of timeslot	Début d'intervalle de temps
End of timeslot	Fin d'intervalle de temps

Figure 82 – Temps d'attente de PDU (PWT)

La durée PWT est configurée par le gestionnaire de système rendant compte de la stabilité intrinsèque des horloges des DLE émettrices et réceptrices, et du temps maximal garanti entre les mises à jour d'horloge.

Par exemple, si les DLE émettrices sont exactes à ±1 ms près par rapport au récepteur dans la période d'expiration d'horloge (dlmo.ClockExpire), il convient que le PWT du récepteur ait une longueur de 2 ms. Une DLE émettrice moins exacte exigera un plus long PWT de récepteur ou une plus courte période d'expiration d'horloge. (Dans cet exemple, il convient que le temps d'écoute de la radio soit légèrement plus long que le PWT de 2 ms, pour rendre compte des durées de SHR et de PHR de la PhPDU.)

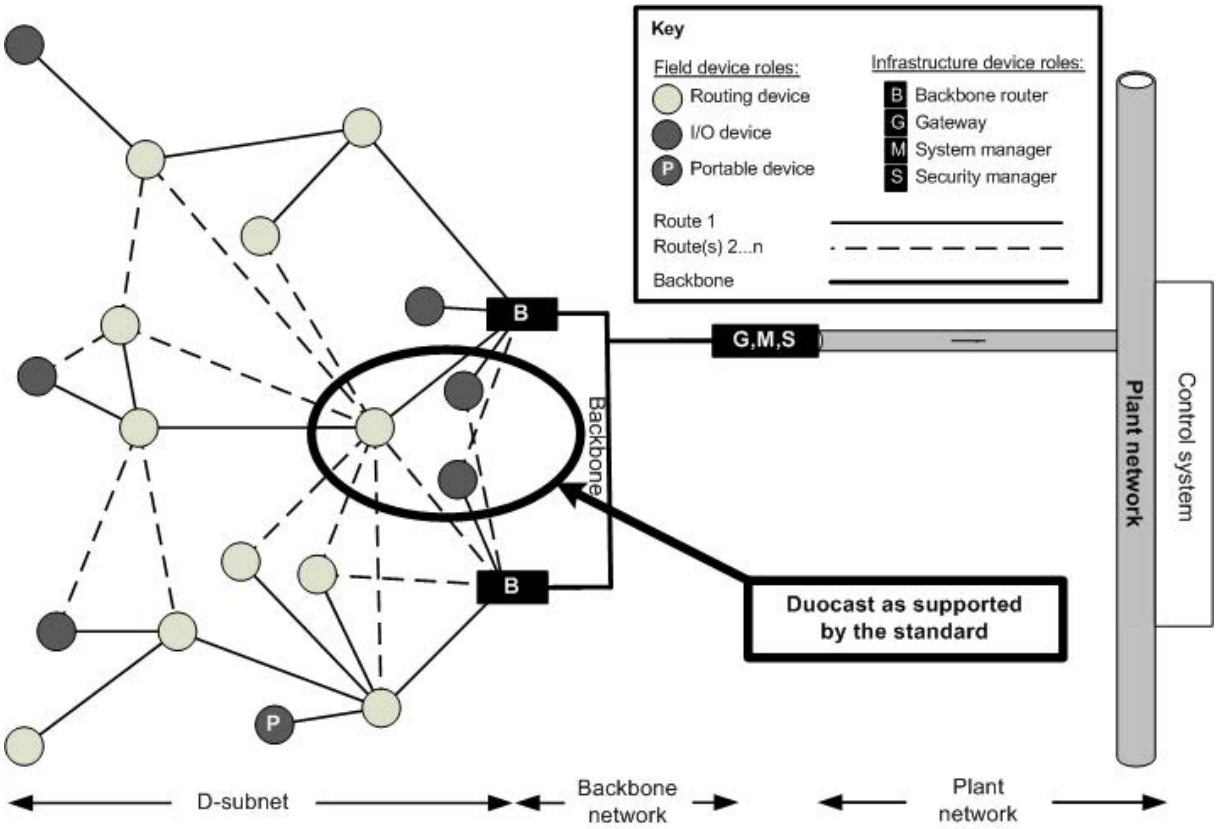
Si, à la fin de son PWT, le récepteur ne commence pas à recevoir la PDU attendue, le récepteur est autorisé à désactiver sa radio pendant la durée de l'intervalle de temps.

Des durées d'intervalle de temps de 10 ms ont un budget de temporisation qui peut allouer environ 2 ms au PWT. Pour un PWT plus long, soit les autres allocations doivent être réajustées, soit la durée d'intervalle de temps doit être augmentée. Par exemple, si le sous-réseau D prend en charge des DLE qui dorment pendant une durée pouvant atteindre deux minutes au maximum, des erreurs de temporisation d'environ ±2 ms peuvent s'accumuler entre les comptes-rendus. Cela peut être pris en charge en augmentant le PWT de (par exemple) 2 ms à 4 ms, avec une augmentation correspondante de la durée d'intervalle de temps et de la consommation d'énergie de récepteur.

NOTE 2 Les DLE avec des intervalles entre rapports peu fréquents sont capables d'être configurées pour vérifier périodiquement le sous-réseau D afin de détecter des DPDU Data recevables. Ces DLE sont capables de recevoir des corrections d'horloge par le sous-en-tête DAUX comme partie intégrante du même processus.

9.1.9.4.7 Transactions en duodiffusion/N-diffusion

Duodiffusion/N-diffusion est une variante de la monodiffusion, dans laquelle un ou plusieurs récepteurs supplémentaires sont programmés pour entendre par hasard les DPDU Data et fournir des acquittements supplémentaires. Duodiffusion/N-diffusion fournit la prise en charge d'accès contrôlée par la latence à une dorsale avec une probabilité accrue de succès au premier essai. Les transactions duodiffusion/N-diffusion sont principalement destinées pour des DLE avec des liaisons à deux ou plus de deux DLE d'infrastructure, en particulier à des routeurs dorsaux (voir la Figure 83).



Légende

Anglais	Français
Key	Légende
Field device roles	Rôles d'appareil de terrain
Routing device	Appareil de routage
I/O device	Appareil E/S
Portable device	Appareil portatif
Infrastructure device roles	Rôles d'appareil d'infrastructure
B Backbone router	B Routeur dorsal
G Gateway	G Passerelle
M System manager	M Gestionnaire de système
S Security manager	S Gestionnaire de sécurité
Route 1	Chemin 1
Route(s) 2…n	Chemin(s) 2…n
Backbone	Dorsale
Control system	Système de commande
Plant network	Réseau d'installation
Duocast as supporte by the standard	Duodiffusion telle que prise en charge par la présente norme
D-subnet	Sous-réseau D
Backbone network	Réseau dorsal

Figure 83 – Prise en charge de duodiffusion dans la norme

Les DLE recevant des acquittements duodiffusion/N-diffusion, cerclées à la Figure 83, ont besoin d'être configurées avec des modèles d'intervalle de temps avec une fenêtre d'écoute étendue pour l'(les) acquittement(s) complémentaire(s). Les DLE envoyant les acquittements duodiffusion/N-diffusion secondaires (l'une des DLE grises cerclées à la Figure 83, par

exemple) ont besoin d'être configurées individuellement avec des modèles d'intervalle de temps avec les fenêtres d'acquittements correctement différées/harmonisées pour leurs acquittements individuels, et avec l'adresse du destinataire primaire prévu pour leur permettre de répondre seulement à la DPDU Data attendue. La prise en charge de la duodiffusion/N-diffusion implique habituellement une plus longue durée d'intervalle de temps d'environ 1 ms à 2 ms par destinataire secondaire, telle que configurée par le gestionnaire de système.

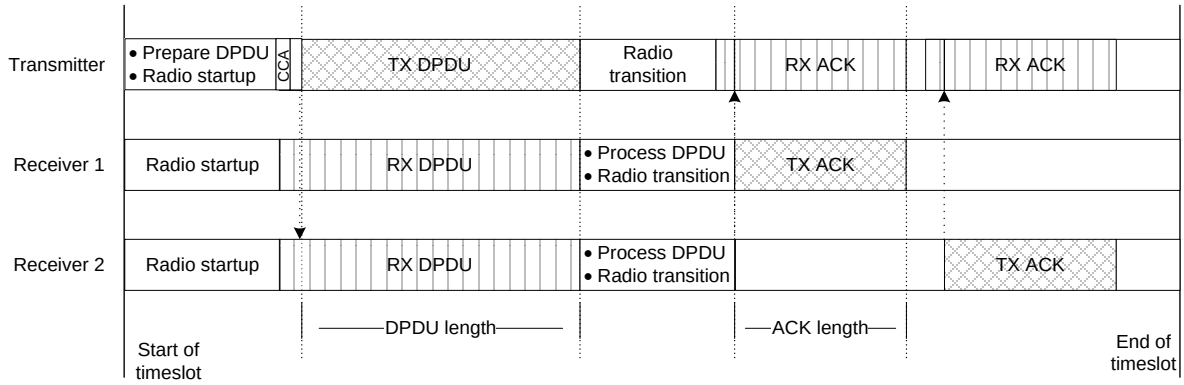
NOTE 1 La coordination de la réponse de duodiffusion/N-diffusion peut impliquer la coordination arrière-voie entre les DLE d'infrastructure répondeuses (car de tels répondeurs sont habituellement des routeurs dorsaux qui sont également connectés à une dorsale de débit plus élevé), mais la coordination par l'intermédiaire de la messagerie de configuration normalisée est également possible, selon la conception des DLE d'infrastructure.

NOTE 2 Si la probabilité de succès d'une seule transaction en monodiffusion acquittée est de p, la probabilité d'échec pour une telle transaction en monodiffusion est de $(1 – p)$. Pour la transaction en duodiffusion correspondante, elle est typiquement de $(1 – p)^2$, alors que dans le cas général pour une transaction en N-diffusion, elle est de $(1 – p)^N$. Ainsi lorsque p = 95 %, la probabilité d'échec est de 5 % pour une transaction en monodiffusion, de 0,25 % pour une duodiffusion, de 0,0125 % pour une transaction en tridiffusion, etc. Ainsi, dans la plupart des environnements relativement planaires, la duodiffusion suffit, mêmes si dans les environnements tridimensionnels fortement métalliques et obstrués tels que les plateformes pétrolières aux larges des côtes (offshore), la tridiffusion peut mériter une prise en considération.

NOTE 3 Dans plusieurs cas, la duodiffusion concerne des contrats avec une taille maximale d'APDU réduite, de telle sorte que les DPDU Data qui initient une transaction ainsi que les sous-baies DPDU et ACK/NAK n'exigent pas une durée supérieure à une transaction en monodiffusion de charge utile maximale avec un acquittement DPDU Data unique et ses acquittements uniques ultérieurs de DPDU ACK/NAK. Dans ce cas, une durée d'intervalle de temps unique est utilisable par les transactions en monodiffusion et les transactions en duodiffusion restreintes, avec toutefois des modèles de transactions différents.

Des intervalles de temps de duodiffusion/N-diffusion peuvent être programmés conjointement avec la largeur de bande numérique disponible pour une répétition de tentative rapide sur une voie alternative (voir Figure 66).

La Figure 84 montre une transaction impliquant l'émission et la réception en duodiffusion.



Légende

Anglais	Français
Transmitter	Emetteur
Prepare DPDU	Préparer DPDU
Radio startup	Démarrage radio
TX DPDU	DPDU émission
Radio transition	Transition radio
RX ACK	ACK réception
Receiver 1	Récepteur 1
RX DPDU	DPDU réception
Process DPDU	Traiter DPDU
TX ACK	ACK émission
Receiver 2	Récepteur 2

Anglais	Français
DPDU length	Longueur DPDU
ACK length	Longueur ACK
Start of timeslot	Début d'intervalle de temps
End of timeslot	Fin d'intervalle de temps

Figure 84 – Transaction en duodiffusion

Dans l'exemple de la Figure 84, la DPDU Data est adressée au récepteur 1, le destinataire primaire, et est entendue par hasard par le récepteur 2, un destinataire secondaire. Pour des transactions en duodiffusion/N-diffusion, l'adresse de destination de la DPDU Data est mise à celle du destinataire primaire (récepteur 1 à la Figure 84), et un acquittement est attendu en provenance d'au moins un destinataire au cours du même intervalle de temps. Comme illustré à la Figure 84, le destinataire primaire émet une DPDU ACK/NAK à la réception de la DPDU Data. Les destinataires secondaires émettent également une DPDU ACK/NAK, mais après un retard dépendant du destinataire afin d'accorder du temps pour permettre à toutes les DPDU ACK/NAK antérieurs de s'achever.

Si une DPDU ACK est reçue d'un quelconque destinataire donnant acquittement, la transaction est achevée et la DPDU Data est supprimée de la file d'attente de messages de la DLE à l'origine de la transaction.

Si une DPDU ACK est reçue d'un destinataire, la DLE à l'origine de la transaction n'est pas tenue de dépenser l'énergie à recevoir et traiter toute DPDU ACK/NAK subséquente. Cependant, il convient que la DLE à l'origine de la transaction vérifie périodiquement qu'elle est capable de recevoir une DPDU ACK/NAK issue de chaque destinataire censé donner acquittement, pour confirmer la disponibilité ininterrompue des récepteurs secondaires.

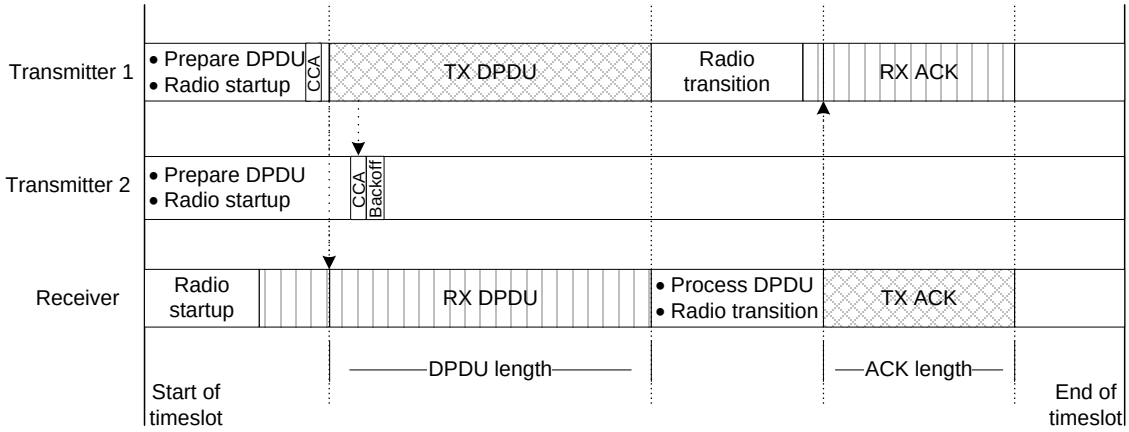
Comme noté en 9.1.5 et décrit en 9.3.4, un acquittement en duodiffusion/N-diffusion issu d'un destinataire secondaire inclut le champ adresse propre de la DLE donnant acquittement dans le champ adresse de source du MHR. Cela permet à l'initiateur de transaction d'identifier la source de l'acquittement et de rapporter épisodiquement celle-ci au gestionnaire de système/sécurité pour faciliter la détection des cyberattaques.

9.1.9.4.8 Intervalles de temps partagés avec CSMA/CA

Les transactions en monodiffusion peuvent se produire dans les intervalles de temps qui sont dédiés à une liaison spécifique. En variante, des intervalles de temps partagés peuvent être désignés pour fournir la largeur de bande à la demande à un ensemble de DLE. Des intervalles de temps partagés sont habituellement configurés pour émettre seulement près du début de l'intervalle de temps.

Les modèles d'intervalle de temps spécifient le temps d'émission sous la forme d'une plage conformément à 9.4.3.3. A tout le moins, le temps d'émission doit être configuré à une plage d'au moins 192 μs, permettant ainsi la prise en charge de ± 96 μs (± 6 périodes symboles PHY) de gigue qui est admissible dans un initiateur de transaction. Si la plage de temps d'émission est configurée pour être supérieure à 200 μs, la DLE doit sélectionner un temps randomisé au sein de la plage pour commencer son émission, faisant de la place raisonnable pour les caractéristiques réelles de gigue d'émission de la DLE.

La Figure 85 montre l'utilisation d'un intervalle de temps partagé avec CSMA/CA.



Légende

Anglais	Français
Transmitter 1	Emetteur 1
Prepare DPDU	Préparer DPDU
Radio startup	Démarrage radio
TX DPDU	DPDU émission
Radio transition	Transition radio
RX ACK	ACK réception
Transmitter 2	Emetteur 2
Backoff	Repli
Receiver	Récepteur
RX DPDU	DPDU réception
Process DPDU	Traiter DPDU
TX ACK	ACK émission
DPDU length	Longueur DPDU
ACK length	Longueur ACK
Start of timeslot	Début d'intervalle de temps
End of timeslot	Fin d'intervalle de temps

Figure 85 – Intervalles de temps partagés avec CSMA/CA active

Dans l'exemple de la Figure 85, deux DLE se disputent l'utilisation de la voie dans un intervalle de temps partagé. Cette approche est utilisée pour tous les intervalles partagés, configurables pour être utilisés de diverses façons.

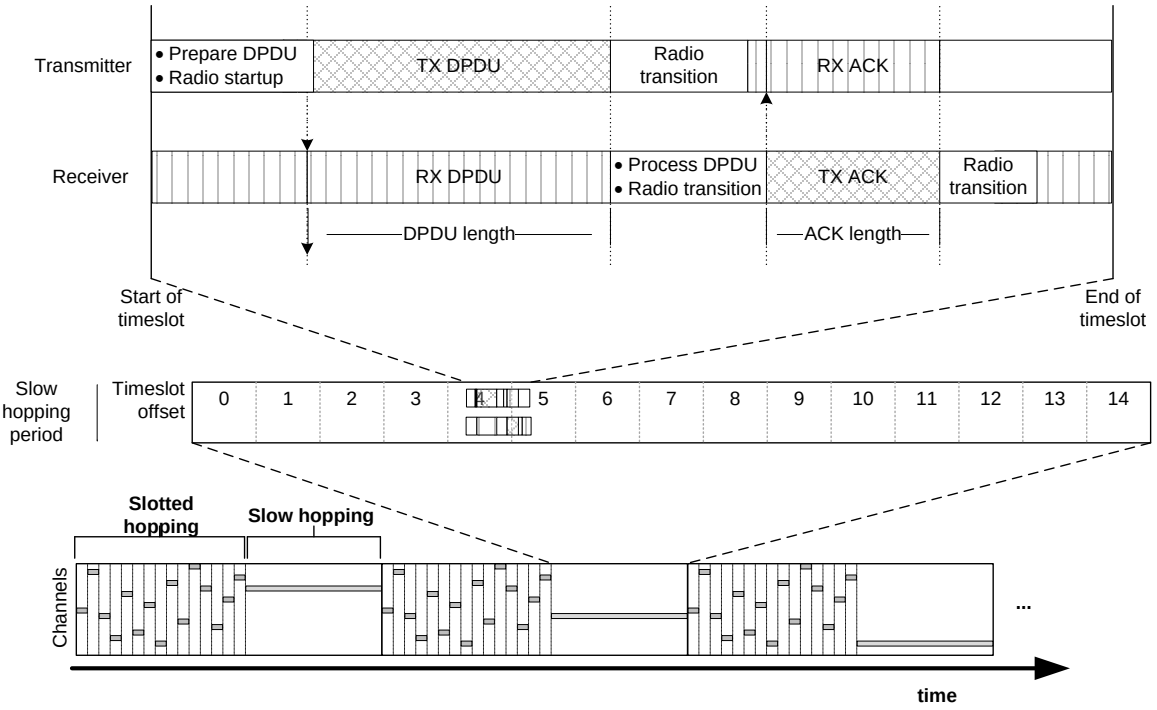
Les priorités au sein d'un intervalle de temps partagé peuvent être gérées en configurant les DLE avec différents modèles d'intervalle de temps. Une DLE avec une DPDU Data de haute priorité, telle qu'une répétition de tentative pour une transaction en duodiffusion ayant échoué, peut être configurée pour émettre sa DPDU Data le plus tôt possible au sein de l'intervalle de temps. Une DLE avec des exigences moins critiques peut être configurée pour retarder son émission à un instant légèrement plus tardif dans l'intervalle de temps, tel que 2 ms plus tard. Si une autre DLE a déjà revendiqué l'intervalle de temps, comme montré à la Figure 85, la CCA de la DLE retardée pourrait (ou pourrait ne pas) détecter que la voie est utilisée et, en conséquence, différer son émission à un autre intervalle de temps.

L'utilisation de CSMA/CA active dans des intervalles de temps partagés peut impliquer des configurations avec de plus longs intervalles de temps et de plus longs temps d'attente de DPDU Data, et l'utilisation de plus d'énergie de récepteur.

9.1.9.4.9 Transactions au cours des périodes de saut de voie lent

Certaines DLE n'ont pas une base de temps suffisamment stable pour communiquer à leur débit de messagerie normal au sein de courts intervalles de temps. Ces DLE peuvent avoir besoin d'utiliser des périodes de saut lent pour leur communication. Les périodes de saut de voie lent sont simplement un jeu d'intervalles de temps concaténés sur la même voie, dans lequel le récepteur fait marcher sa radio sans interruption et l'initiateur de transaction n'est pas tenu de respecter les frontières d'intervalle de temps au sein de la période de saut lent.

Comme montré à la Figure 86, une transaction au cours d'une période de saut de voie lent est très similaire à une transaction en monodiffusion dans un intervalle de temps partagé, excepté que l'émission de DPDU Data peut se produire n'importe où dans la période de saut de voie lent. Les DLE émettrices visent le début d'un intervalle de temps spécifique au sein d'une période de saut de voie lent, selon le propre sens du temps de l'initiateur, qui n'est pas tenu d'être très bien synchronisé avec celui du/des récepteur(s) prévu(s). Les DLE émettrices utilisent la CCA pour vérifier la voie avant émission. En l'absence d'opérations prioritaires de plus haute priorité (telles que la transmission des DPDU Data), un récepteur hébergeant la période de saut de voie lent fait marcher son récepteur radio sans interruption, sauf pour répondre aux DPDU Data qu'il reçoit.



Légende

Anglais	Français
Transmitter	Emetteur
Prepare DPDU	Préparer DPDU
Radio startup	Démarrage radio
TX DPDU	DPDU émission
Radio transition	Transition radio
RX ACK	ACK réception
Receiver	Récepteur
RX DPDU	DPDU réception
Process DPDU	Traiter DPDU
TX ACK	ACK émission

Anglais	Français
DPDU length	Longueur DPDU
ACK length	Longueur ACK
Start of timeslot	Début d'intervalle de temps
End of timeslot	Fin d'intervalle de temps
Slow-hopping period	Période de saut lent
Timeslot offset	Décalage d'intervalle de temps
Channels	Voies
Slotted hopping	Saut discrétisé
Slow hopping	Saut lent
Time	Temps

Figure 86 – Transaction au cours des périodes de saut de voie lent

Les DLE peuvent cibler n'importe quel intervalle de temps au sein d'une période de saut de voie lent, avec une mise en garde majeure: le temps programmé d'intervalle de temps de DPDU Data doit augmenter avec chaque transaction.

Il convient que les DLE respectent les frontières d'intervalle de temps au sein des périodes de saut de voie lent du mieux qu'elles peuvent. Si toutes les DLE au sein des périodes de saut de voie lent sont bien dressées, avec des horloges bien synchronisées, les performances résultant sont approximativement comparables à celles du protocole ALOHA discrétisé.

Une DLE dont le sens de temps diffère de celui de ses récepteurs prévus émettra nominalement près du début d'un intervalle de temps particulier, selon son propre sens de temps. Cependant, une telle DLE peut réellement initier l'émission dans n'importe quelle phase de n'importe quel intervalle de temps au sein de la période de saut de voie lent.

Par exemple, une DLE qui a une source d'horloge avec une stabilité de $\pm 50 \times 10^{-6}$ sur un intervalle de quelques minutes, en raison de fluctuations environnementales non compensées, peut dormir plus de 3 min entre les transactions, ce qui correspond à une dérive d'horloge de ±9 ms environ. Une DLE de ce type pourrait attendre une annonce programmée pour se faire resynchroniser avant l'émission. En variante, elle pourrait émettre au cours d'une période de saut de voie lent (si disponible) et recevoir la synchronisation de temps dans l'acquittement. En continuant cet exemple, une DLE avec une exactitude de ±9 ms sélectionnerait un des intervalles de temps au sein de la période de saut de voie lent, et émettrait en utilisant le modèle approprié de liaison. La DLE émettrait nominalement dans un intervalle de temps particulier, selon le propre sens du temps de la DLE, mais peut réellement émettre dans un intervalle de temps différent.

Dans cet exemple, il convient que la DLE ne tente pas d'émettre dans le premier ou le dernier intervalle de temps dans la période de saut de voie lent, car elle peut réellement émettre à l'extérieur de la plage de temps disponible. Il est de la responsabilité de la DLE d'éviter de sélectionner des intervalles de temps incompatibles avec les capacités de chronométrage de la DLE, rendant compte de la propre dérive d'horloge non étalonnée de la DLE en combinaison avec la dérive d'horloge de 10×10^{-6} autorisée pour un routeur voisin ou la dérive d'horloge de 100×10^{-6} autorisée pour un appareil de terrain voisin autre que de routage.

Toutes les DPDU Data dans des périodes de saut de voie lent doivent inclure un octet supplémentaire dans l'en-tête de DPDU Data pour indiquer quel décalage d'intervalle de temps au sein de la période de saut de voie lent était prévu. Cela permet à la DLE réceptrice de reconstruire les informations relatives aux intervalles de temps issues de la DLE émettrice, d'appliquer la correction de temps à travers les intervalles de temps, de valider l'exactitude de l'horloge de l'initiateur de transaction, et d'utiliser le temps de début programmé de l'intervalle

de temps comme support de sécurité pour l'authentification de message. (Cela est spécifié dans les sous-en-têtes Media Access Control de DPDU Data, DMXHR; voir 9.3.3.4.)

S'il y a lieu, un décalage corrigé d'intervalle de temps est fourni dans l'acquittement (voir 9.3.4). L'identification non ambiguë d'intervalle de temps partagé est nécessaire tant à l'initiateur de transaction qu'aux récepteurs de transaction pour authentifier la DPDU Data et resynchroniser le temps. Même si la DLE émettrice a un sens de temps hautement exact, les DLE réceptrices pourraient ne pas en avoir. Par conséquent, l'octet de décalage de saut de voie est exigé pour toutes les DPDU Data utilisant une supertrame de saut de voie lent.

L'intervalle de temps initial au sein d'une période de saut de voie lent est défini comme ayant un décalage de zéro.

9.1.10 Adressage de sous-réseau D

9.1.10.1 Types d'adresse

Les adresses DL16Addresses doivent toujours être utilisées au sein d'un sous-réseau D, excepté que les adresses EUI64Addresses sont utilisées d'une manière limitée au cours de la phase initiale du processus de rattachement pour communiquer avec l'appareil se rattachant.

Chaque DLE dans un sous-réseau D est identifiée de trois façons:

- Chaque DLE de sous-réseau D a un identificateur d'adresse EUI64Address qui est présumé être unique.
- Chaque DLE conforme à la présente norme doit avoir au moins une adresse IPv6Address assignée lorsqu'elle rejoint le sous-réseau D. Cependant, au sein de la DL, seul le pseudonyme (alias) DL16Address pour cette adresse IPv6Address (décrite ci-après) doit être utilisé.
- Chaque DLE ou appareil étranger qui est accessible à travers un sous-réseau D a une adresse DL16Address unique dans un sous-réseau D, qui est un pseudonyme pour son adresse IPv6Address. La portée de toute adresse DL16Address doit être limitée à un sous-réseau D particulier.

L'adresse EUI64Address doit être utilisée comme une nouvelle adresse de DLE pour l'adressage de voisin immédiat avant et pendant le processus de rattachement. Une fois qu'une DLE a rejoint le sous-réseau D et reçu son adresse IPv6Address, elle doit être adressée soit par son adresse IPv6Address, soit par son pseudonyme DL16Address local à un sous-réseau D pour l'adresse IPv6Address en question.

L'adresse EUI64Address est également utilisée dans chaque nonce de sécurité de la DPDU. Chaque fois qu'une adresse DL16Address est utilisée dans une DPDU Data ou une DPDU ACK/NAK, le(s) destinataire(s) de la DPDU a(ont) besoin d'une connaissance a priori de l'adresse EUI64Address correspondante. Ces informations relatives aux voisins sont fournies par le gestionnaire de système comme partie intégrante du processus d'établissement de liaison. Une exception est faite pour une nouvelle DLE qui communique avec un voisin qui a annoncé son adresseDL16Address. Dans ce cas, la DLE de la DLE se joignant doit acquérir l'adresse EUI64Address de la DLE annonciatrice en initiant une transaction avec ce voisin, lui envoyant une DPDU Data avec une charge utile vide tout en demandant l'adresse EUI64Address dans la DPDU ACK/NAK de réponse, comme décrit en 9.3.3.3. Pour cette transaction d'amorçage, la clé D applicable est K_global, utilisée avec un DMIC-32, qui est la même que pour d'autres DPDU Data impliquant une adresse EUI64Address. (Voir 9.1.11 pour un débat relatif à la sécurité de DL.)

Lorsqu'une adresse DL16Address se réfère à une DLE au sein d'un sous-réseau D, l'adresse DL16Address est toujours l'adresse MAC 16 bits de la DLE. Lorsqu'une DPDU Data contient une adresse DL16Address qui se réfère à un appareil hors de la portée du sous-réseau D, la DPDU Data est acheminée vers une DLE qui est en service comme routeur dorsal, qui effectue un mapping de l'adresse DL16Address à sa contrepartie d'adresse IPv6Address.

La plupart des DPDU Data incluent deux adresses DL16Addresses de source et deux adresses DL16Addresses de destination. Une paire d'adresses DL16Address source/destination est pour l'adressage de prochain saut à la sous-couche MAC. Une deuxième paire d'adresses DL16Address source/destination (des pseudonymes D, en fait) est au sein du sous-en-tête DADDR de la DPDU Data, où elles spécifient les NLE source et destination finales locales à un sous-réseau D via leurs pseudonymes D.

Le routage est habituellement spécifié par des graphes, pas par des adresses D. Cependant, lorsque le routage par source est utilisé au sein d'un sous-réseau D, des adresses DL16Address sont normalement utilisées. Encore une fois, la seule exception est que les adresses EUI64Address sont utilisées pendant le processus de rattachement pour la communication avec un voisin immédiat au sein du sous-réseau D.

9.1.10.2 Identificateur de sous-réseau et unicité des adresses DL16Address

La portée d'une adresse DL16Address est un sous-réseau D. Un sous-réseau D voisin peut utiliser la même adresse DL16Address pour référencer une DLE différente. Des sous-réseaux D différents peuvent utiliser des adresses DL16Address différentes pour se référer au même appareil de dorsale.

Chaque sous-réseau D a un identificateur de sous-réseau D de 16 bits (dlmo.SubnetID; voir 9.4.2.1), qui a la même valeur que le PAN ID situé dans l'en-tête de MPDU de l'IEEE 802.15.4. Dans la portée d'un réseau, chaque sous-réseau D doit avoir un dlmo.SubnetID unique. Au sein de la portée d'un réseau, chaque sous-réseau D doit avoir une dlmo.SubnetID unique. Cependant, des réseaux différents ne sont pas nécessairement coordonnés et, donc, il n'est pas garanti que les dlmo.SubnetID à travers les réseaux conformes à la norme contigus sont uniques. Ainsi, il est possible, bien que peu probable, qu'une DPDU issue d'un différent sous-réseau D conforme à la norme soit reçue avec ce qui semble être une adresse DL16Address valide et un PAN ID valide. La DLE se fonde sur le DSC pour rejeter de telles DPDU à la réception en raison d'une discordance de DMIC ou d'une non-authentification de DMIC, discordance due à des clés symétriques de sécurité D non identiques.

SubnetID=0x0000 et SubnetID=0xFFFF ne doivent pas être utilisés comme ID de sous-réseau D dans la présente norme. SubnetID=0x0001 est réservé pour les sous-réseaux D de configuration (voir 13.1) et ne doit pas être autrement utilisé.

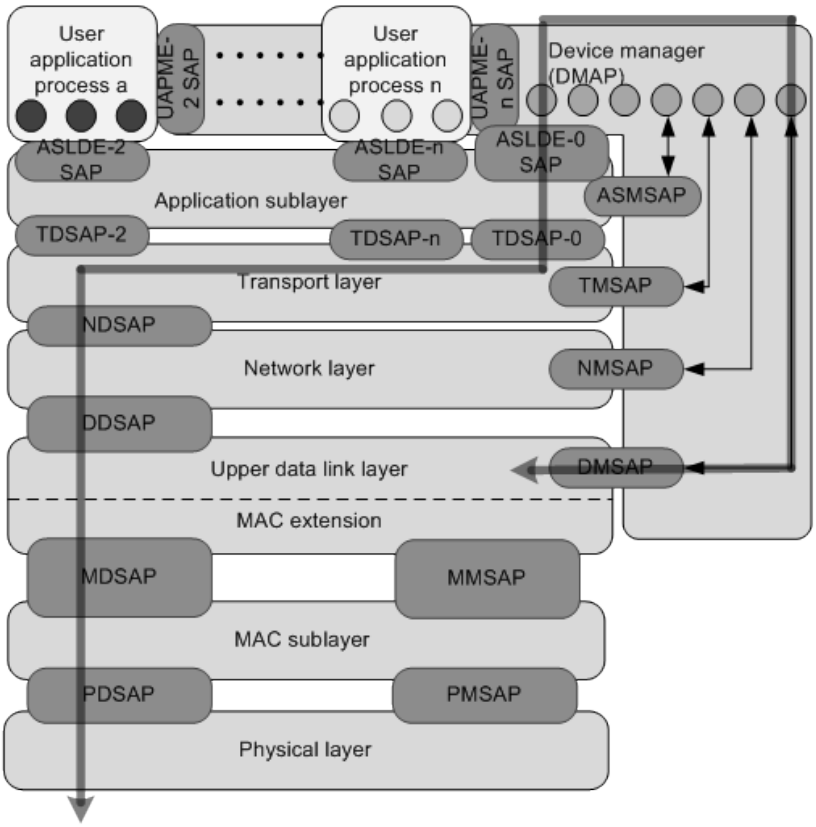
9.1.11 Service gestion de DL

9.1.11.1 Généralités

Des messages de gestion vers une DLE sont entièrement sécurisés à l'AL, en utilisant la relation de bout en bout entre le gestionnaire de système/sécurité et le DMAP de la DLE.

La MAC de DL basée sur l'IEEE 802.15.4 n'est pas accessible directement par le DMAP, elle est à la place configurée indirectement à travers la DMSAP. Cela isole le reste de la présente norme par rapport aux changements évolutifs apportés aux spécifications de l'IEEE, facilite la future adoption de spécifications de couche physique de remplacement (par exemple, radios), qui peuvent avoir des MAC associées alternatives ou améliorées, et permet que certains aspects opérationnels de MAC et de PHY (tels que CCA) soient utilisés ou pas, sur une base d'intervalle de temps par intervalle de temps.

Comme montré à la Figure 87, la gestion de DL commande généralement le flux à travers la pleine pile de protocoles de communication définie par la présente norme.



Légende

Anglais	Français
User application process a	Processus d'application utilisateur a
User application process n	Processus d'application utilisateur n
Device manager (DMAP)	Gestionnaire d'appareil (DMAP)
Application sub-layer	Sous-couche d'application
Transport layer	Couche transport
Network layer	Couche réseau
Upper data link layer	Couche liaison de données supérieure
MAC extension	Extension MAC
MAC sub-layer	Sous-couche MAC
Physical layer	Couche physique

Figure 87 – Flux de SAP de gestion de DL à travers une suite normalisée de protocoles

Une DLE est configurée par l'intermédiaire de son DMAP par le biais de DMSAP. La plupart des SAP de gestion sont génériques, lisant et établissant simplement des structures de données au sein de la couche. Ces structures de données définissent généralement comment une DLE exploite son diagramme d'états. Le DMAP communique avec le gestionnaire de système par la sous-couche d'application, en utilisant la sécurité de bout en bout.

Pour des informations relatives au traitement général des objets de gestion normalisés, voir 6.2.5 et 6.2.6.

9.1.11.2 Attributs de gestion et attributs indexés

Les DMSAP impliquent la manipulation de l'objet de gestion de DL (DLMO). Le DLMO inclut une diversité d'attributs qui sont utilisés pour configurer la DLE et/ou rapporter son statut.

Certains attributs de DLMO s'appliquent à des valeurs spécifiques. Par exemple, l'attribut dlmo.SubnetID fournit l'ID de sous-réseau pour le sous-réseau D que la DLE a rejoint.

Certains attributs de DLMO peuvent être visualisés sous forme de tableaux avec un ensemble de rangées à indices. Chaque attribut de ce type est spécifié comme un OctetString indexé. Par exemple, la DLE inclut l'attribut dlmo.Neighbor, qui est un attribut OctetString indexé contenant un ensemble de DLE voisines. Chaque entrée de l'attribut dlmo.Neighbor inclut un jeu de champs pour le voisin en question, tel qu'un indicateur signalant si ce voisin est une source d'horloge de DL ou non. Chaque entrée de la table dlmo.Neighbor est identifiée de façon univoque par l'adresse DL16Address du voisin. Les attributs OctetString indexés dans le DLMO comprennent:

- dlmo.Ch: modèles de saut de voie;
- dlmo.TsTemplate: modèles d'intervalle de temps;
- dlmo.Neighbor: voisins de DLE;
- dlmo.NeighborDiag: diagnostics pour voisins de DLE;
- dlmo.Graph: graphes pour routage;
- dlmo.Superframe: supertrames spécifiant des attributs communs pour clés associées;
- dlmo.Link: liaisons, dont chacune est associée à une supertrame;
- dlmo.Route: chemins utilisables dans des sous-en-têtes DROUT.

Les relations entre ces attributs sont décrites en 9.4.3.1.

Les attributs de DLMO doivent être maintenus à travers le DMAP en utilisant les méthodes qui sont décrites à l'Article 9 et à l'Article 12. Les services Read et Write de l'ASLE décrits dans le Tableau 271 fournissent un cadre général pour lire et écrire des attributs de DLMO. Les méthodes pour écrire des attributs qui doivent devenir actifs au temps TAI de basculement, ainsi que les méthodes pour lire et écrire des attributs OctetString indexés, sont décrites en 9.5.

Le format des attributs de DLMO est défini dans la spécification de la DLE. Lorsque ces objets sont intégrés au sein de messages par liaison radio, les formats spécifiés doivent être utilisés. La présente norme ne contraint pas (et ne peut pas contraindre) la façon dont les données sont stockées au sein d'une DLE, ou définir comment les informations correspondantes sont relayées en internes à une DLE.

9.1.11.3 Messages de gestion issus de voisins immédiats

Toute DPDU Data peut inclure un sous-en-tête DAUX, par exemple transportant des informations d'annonce issues d'un voisin immédiat. Les informations de sous-en-tête DAUX ne sont pas propagées à des couches supérieures de la suite de protocoles de communication. Dans la plupart des cas, le contenu du sous-en-tête DAUX est destiné au processus de gestion des DLE de destinataires, qui à son tour peut utiliser ces informations pour configurer le diagramme d'états de la DLE. Par exemple, le sous-en-tête DAUX peut inclure des définitions de supertrame qui sont prévues être utilisées par la DLE comme point de départ dans le processus de rattachement et/ou pour la découverte de voisins.

Le sous-en-tête DAUX est modélisé comme étant instantanément visible au DMAP et immédiatement soumis à une action s'il y a lieu.

NOTE Sachant que les normes s'appliquent seulement aux aspects visibles extérieurement, l'interface DMSAP interne n'est pas soumise à normalisation.

9.1.11.4 Plusieurs sous-réseaux D

Les objets de gestion de DL prennent en charge un seul sous-réseau D actif à la fois. Toutes les adresses DL16Addresses doivent être uniques dans la portée du seul sous-réseau D en

question. Cette contrainte n'empêche pas un appareil de participer simultanément à plusieurs sous-réseaux D. Plusieurs sous-réseaux D pourraient être modélisés comme plusieurs instances d'une DLE, mais une telle opération n'est pas spécifiée par la présente norme.

Si dlmo.SubnetID=0, la DLE ne participe pas encore au sous-réseau D.

9.1.11.5 Plusieurs PhLE (radios)

Chaque DSAP d'une DLE est supposé être associé à une seule PhLE (radio). Cela n'empêche pas des mises en œuvre avec plusieurs radios, qui pourraient être modélisées comme plusieurs instances d'une DLE. Une telle opération n'est pas spécifiée par la présente norme.

9.1.12 Relation entre DLE et DSC

La relation entre la DLE et le DSC est décrite en 7.3.2. Une revue soignée de 7.3.2 est essentiel pour quiconque qui souhaite comprendre complètement le fonctionnement de la DLE.

Généralement, la DLE s'appuie sur le DSC pour l'authentification, l'intégrité et la confidentialité conditionnelle. Toutes les DPDU incluent un DMIC qui valide sans ambiguïté qu'une MPDU a comme source une DLE qui a partagé la même clé de sous-réseau D, vs. celles qui mettent en œuvre un autre protocole utilisant une modulation, un codage et des voies similaires. Le DMIC utilise une clé de sécurité non secrète pendant le processus de rattachement; mais il est habituellement configuré pour utiliser une clé secrète partagée une fois que la DLE s'est rattachée.

Les DPDU Data ACK/NAK sont étendues par la DLE, en insérant le DMIC de la DPDU dans l'en-tête DPDU ACK/NAK, comme un champ virtuel avant d'être traitées par le DSC. Ce champ virtuel, qui n'est pas émis, est inclus dans le calcul du DMIC de la DPDU ACK/NAK. Ainsi, le DMIC de la DPDU Data est essentiellement repris en écho dans la DPDU ACK/NAK sans émettre réellement le champ. De cette manière, la DPDU ACK/NAK est liée sans ambiguïté aux DPDU Data correspondantes de la transaction D.

9.1.13 Découverte de voisins de DLE

9.1.13.1 Généralités

Les sous-réseaux D sans fil conformes à la présente norme sont détectés par la DLE. Une nouvelle DLE peut entendre les annonces issues de DLE voisines qui ont déjà rejoint un sous-réseau D d'intérêt. Cela s'appelle la découverte de voisins. A la suite de la découverte de voisins, la DLE peut rejoindre le sous-réseau D comme décrit en 7.4.

Les processus d'annonce de DL et de découverte de voisins de DL sont les blocs modules de base qui permettent à une DLE d'apprendre les adresses DL16Address et EUI64Address d'un routeur voisin, et le jeu de liaisons programmées qui ont été allouées pour être utilisées lors du rattachement. Ces informations sont ensuite utilisées par la DLE et la DME pour rejoindre le sous-réseau D.

Les DLE découvrent les sous-réseaux D par l'intermédiaire des DPDU Data d'annonce reçues en provenance d'une ou plusieurs DLE d'annonce qui annoncent périodiquement leur présence. Une DLE d'annonce est capable d'agir comme un proxy dans le processus de configuration ou de rattachement. Une annonce contient des informations qui permettent à une nouvelle DLE d'envoyer une demande de rattachement à la DLE d'annonce, pour la relayer à un gestionnaire de système, et de recevoir un certain temps plus tard une réponse de rattachement.

Après avoir rejoint un sous-réseau D, la DLE reçoit des DPDU Data d'annonce issues de DLE voisines dans le sous-réseau D et dresse une liste locale de voisins candidats avec lesquels elle peut avoir des communications de qualité raisonnable. Cette liste de candidats est

rapportée à un gestionnaire de système par l'intermédiaire de l'attribut dlmo.Candidates. Le gestionnaire de système utilise ces informations pour déterminer comment les DLE s'adaptent dans la topologie de sous-réseau D. A leur tour, ces informations sont utilisées pour établir des relations de communication entre la nouvelle DLE et ses voisins.

Les DLE d'annonce peuvent être découvertes en utilisant le balayage passif, le balayage actif, ou une combinaison de balayage passif et actif.

Dans le balayage passif, la DLE se met périodiquement à l'écoute pour détecter des annonces sur une série de voies. En général, une DLE de balayage passif alimentée par batterie se mettra fréquemment à l'écoute lorsqu'elle est mise sous tension pour la première fois. Si un sous-réseau D n'est pas découvert rapidement, la DLE peut prolonger la durée de vie de sa batterie en balayant moins fréquemment et/ou en allouant des intervalles de temps plus courts pour chaque balayage. De telles réductions peuvent conduire à des retards substantiels dans le rattachement au sous-réseau D et/ou de formation de sous-réseau D, et ne sont donc utilisées que si la stratégie de rattachement rapide a échoué.

Le balayage actif surmonte certains inconvénients du balayage passif. Les DLE qui sont configurées pour le balayage actif recherchent un sous-réseau D en émettant périodiquement des DPDU de sollicitations, qui déclenchent en réponse les DPDU Data de sollicitations issues de routeurs voisins. La DLE émettant la DPDU Data de sollicitation est appelée un interrogateur de balayage actif, alors que la DLE répondeuse est appelée un hôte de balayage actif. Les hôtes de balayage actif dépensent de l'énergie à faire fonctionner leurs récepteurs radio pendant l'écoute pour détecter des DPDU Data de sollicitation. Certains hôtes de balayage actif ont de l'énergie disponible pour le fonctionnement continu de récepteur. Des hôtes de balayage actifs avec des sources d'énergie plus limitées peuvent être configurés pour écouter sans interruption pendant certaines périodes de temps, comme pendant la formation de sous-réseau D.

Des programmes de liaison utilisés pour le rattachement ne sont pas nécessairement liés à des programmes de supertrame utilisés pour le fonctionnement normal du sous-réseau D. Ainsi, peu d'informations relatives au fonctionnement du sous-réseau D ont besoin d'être acheminées dans les annonces.

9.1.13.2 Sous-en-tête auxiliaire et annonces

Les annonces et les sollicitations de DL sont acheminées dans un sous-en-tête auxiliaire de DPDU Data (DAUX) au sein de l'en-tête de DPDU (DHR). Voir 9.3.1 pour une vue d'ensemble du DHR. Une DPDU Data qui achemine une sollicitation est appelée une DPDU de sollicitation. De même, une DPDU Data qui achemine une annonce est appelée une DPDU d'annonce.

Le sous-en-tête DAUX est habituellement absent d'un DHR, mais doit être inclus dans tout DHR s'il est configuré ainsi pour liaison particulière. Une DPDU Data contenant un sous-en-tête DAUX peut également transporter une charge utile de couche supérieure qui est sans rapport avec la fonction de découverte de voisins.

L'émission d'une annonce est déclenchée par un fanion d'annonce tel que configuré au sein d'une définition de liaison. Le fanion d'annonce indique qu'une annonce doit être émise dans un intervalle de temps de supertrame, en l'absence d'une liaison de priorité plus haute.

La même définition de liaison peut également inclure un fanion d'émission, auquel cas la DLE doit vérifier la file d'attente de messages pour déceler des DPDU Data sortantes concordantes. Ainsi, la DPDU Data peut transporter simultanément:

- une annonce; et
- une charge utile de DSDU qui est complètement sans rapport avec l'annonce.

La capacité de charge utile de la DPDU Data est réduite lorsqu'une annonce est intégrée au sein du DHR. Certaines DPDU Data dans la file d'attente de messages, en particulier les messages qui ont été fragmentés à la NL, peuvent être trop longues pour être combinées avec une annonce. De tels messages ne sont pas des candidats pour les liaisons qui sont partagées avec une annonce, donnant effectivement à l'annonce l'accès prioritaire à ces intervalles de temps.

NOTE Lorsque des messages sont fragmentés par la NL, la taille du fragment est établie par la NL avant d'être passée à la DLE, la taille de fragment étant configurée par le gestionnaire de système. Dans les configurations où des annonces sont rarement combinées à des liaison d'émission, la taille maximale du fragment est souvent limitée par la capacité de charge utile de la DPDU Data sans considérer l'annonce combinée.

En général, les DPDU Data contenant une annonce utilisent un DMIC basé sur la clé de sécurité du sous-réseau D, fournissant ainsi une annonce à laquelle les DLE peuvent faire confiance après qu'elles ont rejoint le sous-réseau D. Ce DMIC ne peut pas être validé avant le rattachement, car une DLE non rattachée n'a pas encore la clé de sécurité de sous-réseau D. Par conséquent, une DLE non rattachée qui balaye pour trouver un sous-réseau D est autorisée à traiter une annonce dans un sous-en-tête DAUX, même si elle est incapable d'authentifier la DPDU Data contenant le DAUX.

Sans la garantie d'un DMIC, une DLE non rattachée recevant une annonce peut toujours utiliser la FCS de l'IEEE 802.15.4 comme contrôle d'intégrité. Cependant, la FCS de l'IEEE 802.15.4 seule ne filtre pas les MPDU issues d'autres systèmes qui utilisent l'EEE 802.15.4. Par conséquent la présente norme fournit un contrôle d'intégrité supplémentaire, n'impliquant pas de clé de sécurité, couvrant spécifiquement le sous-en-tête d'annonce, telle que spécifiée en 9.3.5.2.4.3.

9.1.13.3 Sollicitation de balayage actif et réponse

Dans le balayage actif, une nouvelle DLE, agissant comme un interrogateur de balayage actif, sollicite périodiquement des annonces issues d'hôtes qui se trouvent être dans la plage radio. Une DLE recevant la sollicitation peut être configurée pour répondre avec une annonce.

Le balayage actif est destiné pour les configurations dans lesquelles une DLE d'annonce, en sa qualité d'hôte de balayage actif, est capable faire fonctionner son récepteur de façon plus ou moins continue dans une configuration de saut de voie lent. Des hôtes de balayage actif peuvent être alimentés en puissance sans interruption. En variante, ils peuvent être des DLE à énergie contrainte qui font marcher leurs récepteurs dans une configuration de saut de voie lent pendant des périodes limitées, comme pendant la formation de sous-réseau D.

La demande de sollicitation est codée dans le sous-en-tête DAUX de la DPDU Data de sollicitation. Une DPDU Data de sollicitation ne doit pas contenir une charge utile de NL.

Une DPDU Data de sollicitation, lorsqu'elle est reçue par une DLE hôtesse de balayage actif, amène cette DLE à une DPDU Data d'annonce dans le prochain intervalle de temps si la DLE est ainsi configurée. Un routeur recevant une DPDU Data de sollicitation doit répondre en émettant une DPDU Data d'annonce dans le prochain plein intervalle de temps si, et seulement si:

- la liaison de réception par défaut pour le balayage (Tableau 165) s'applique dans le prochain intervalle de temps et se produit sur la même voie radio que la sollicitation; et
- l'attribut dlmo.ActScanHostFract de la DLE est configuré pour la réponse à des DPDU Data de sollicitation. L'attribut dlmo.ActScanHostFract indique que la fraction du temps dans laquelle il convient que la DLE réponde lorsqu'elle a reçu un sollicitation de balayage actif, où
  - une valeur de 0 indique que la DLE n'est pas configurée comme un hôte de balayage actif et elle ne répondra pas à des sollicitations,
  - une valeur de 255 indique qu'il convient que la DLE réponde toujours aux sollicitations, et

  - une valeur dans la plage 1..254 indique que la DLE fait une sélection uniformément aléatoire dans la plage 1..255 chaque fois qu'elle reçoit une sollicitation, et ne répond pas avec une sollicitation si le résultat est supérieur à dlmo.ActScanHostFract, produisant ainsi une réponse de sollicitation de DPDU Data avec la probabilité dlmo.ActScanHostFract/255; et

- la DLE est configurée pour répondre à l'identificateur de sous-réseau D inclus dans la DPDU Data d'annonce, comme décrit en 9.4.2.20. Une DPDU Data de sollicitation peut être configurée pour inclure un identificateur de sous-réseau D pour limiter les répondants à un jeu désiré de sous-réseaux D.

Le prochain plein intervalle de temps, dans ce contexte, doit être défini comme le prochain intervalle de temps qui commence à la suite de la fin de PhPDU de la sollicitation plus 1 ms. Dans le prochain intervalle de temps en question, la DPDU Data d'annonce doit être émise en utilisant la temporisation telle que définie dans le modèle d'initiateur de transaction par défaut dans le Tableau 166, même si le modèle par défaut en question est neutralisé pendant la configuration de la DLE.

Il convient qu'un interrogateur de balayage actif, qui n'est pas présumé synchronisé avec la temporisation de sous-réseau D, permette à sa radio de recevoir une DPDU Data d'annonce dès 3 212 μs après la fin de la DPDU Data de sollicitation, qui est 1 ms plus les 2 212 μs du Tableau 166. Suivant cet instant, il convient que l'interrogateur de balayage actif maintienne son récepteur activé assez longtemps pour recevoir une DPDU Data d'annonce commençant à n'importe quel instant pendant une pleine durée d'intervalle de temps. Il convient que l'interrogateur de balayage actif utilise sa propre durée d'intervalle de temps comme la durée d'intervalle de temps supposée de l'hôte de balayage actif. Par conséquent, lorsque des DPDU Data de sollicitations sont utilisées, il convient que l'interrogateur de balayage actif soit configuré avec une durée d'intervalle de temps qui concorde ou dépasse la durée d'intervalle de temps du sous-réseau D cible.

Une DPDU Data de sollicitation est exigée pour une liaison qui est configurée comme une liaison de sollicitation.

Pour prendre en charge le balayage passif par de nouvelles DLE, des hôtes de balayage actif peuvent également être configurés pour émettre périodiquement des DPDU Data d'annonce.

Il peut être nécessaire de supprimer des sollicitations, pour rendre compte de situations où il est dangereux ou illégitime qu'une DLE fasse fonctionner son émetteur radio sans autorisation. Pour traiter de telles situations, l'attribut dlmo.RadioSilence de DLMO fournit un mécanisme pour désactiver les sollicitations avec toutes les autres émissions de DLE.

Les sollicitations sont désactivées par défaut. Les sollicitations ne sont pas utilisées dans la configuration par défaut, et dlmo.ActScanHostFract a zéro comme valeur par défaut.

9.1.13.4 Balayage continu

Il convient que la découverte de voisins soit un processus continu, même après qu'une DLE a rejoint le sous-réseau D. Les balayages continus peuvent, au fil du temps, aider à former un sous-réseau D plus optimal. Les routeurs alimentés électriquement par le secteur qui dépensent une partie substantielle de leur temps à écouter peuvent, au fil du temps, recevoir de nombreuses annonces issues de routeurs voisins. Les appareils alimentés par batterie à piles ne peuvent pas dépenser un pourcentage élevé de leur temps à écouter, mais même s'ils échantillonnent juste la voie périodiquement, ils peuvent, sur des périodes de temps prolongées, construire une image des routeurs voisins. Si le sous-réseau D est configuré avec un programme coordonné d'annonces, un tel balayage peut être accompli plus efficacement.

Une fonction de gestion de système peut établir un programme global de sous-réseau D pour des annonces, et une DLE rattachée peut être configurée avec un programme de liaisons de réception pour s'assurer que de telles annonces sont entendues au fil du temps. Lorsqu'elle

est utilisée, cette approche permet à des DLE sur le sous-réseau D de trouver efficacement des voisins. En plus, une DLE de faible facteur d'utilisation peut être configurée pour utiliser ces annonces programmées pour rester synchronisée en temps avec le sous-réseau D.

Les annonces sont authentifiées avec un DMIC, pour permettre à des DLE qui ont rejoint le sous-réseau D de s'appuyer de façon fiable sur des annonces programmées comme une source digne de confiance d'informations relatives à la temporisation et à la connectivité.

9.1.14 Découverte de voisins et rattachement – considérations de DL

9.1.14.1 Généralités

La DL fournit un mécanisme configurable pour découvrir des DLE voisines.

Pendant la configuration, une DLE est configurée pour rechercher par balayage des voisins qui peuvent agir comme des proxys dans le processus de rattachement. Lorsqu'une annonce est reçue en provenance d'un voisin candidat, la DLE utilise les informations contenues dans l'annonce pour créer des liaisons de communication à destination et en provenance du voisin en question et fournit ensuite au DMAP les informations d'adressage du voisin. Pour le reste du processus de rattachement, la DLE fournit la prise en charge des communications aux couches supérieures en passant les DPDU Data à destination et en provenance du voisin.

Après avoir rejoint le sous-réseau D, la DLE est chargée, implicitement ou explicitement, par le gestionnaire de système de rechercher par balayage de nouveaux voisins pendant une période de temps désignée, en utilisant des supertrames et des liaisons fournies par le gestionnaire de système. Le résultat de ce balayage est rapporté comme informations relatives aux voisins au gestionnaire de système qui utilise les informations pour pourvoir à la DLE une configuration optimale au sein de la maille de DLE. La DLE continue alors d'accumuler des informations relatives aux nouveaux voisins candidats dans tout son cycle de vie, et ces informations sont périodiquement rapportées au gestionnaire de système pour faciliter la configuration de configurations de maillage améliorées et adaptatives.

Le processus de rattachement au sous-réseau D, du point de vue de la DLE, peut être récapitulé à titre informatif comme suit:

– La DLE, possiblement dans son état d'usine, recherche une annonce auprès de la DLE de configuration. Cette recherche est intégrée dans la DLE telle que définie par la norme.
– La DLE reçoit une annonce issue de la DLE de configuration. Cette annonce, avec l'ID de sous-réseau D = 1, fournit une configuration comprimée, mais pleinement fonctionnelle, pour le diagramme d'états de la DLE. La DLE utilise cette configuration pour communiquer avec la DLE de configuration. Pendant le processus de configuration, une configuration de DLE différente, qui est ultérieurement utilisée pour rechercher un sous-réseau D cible, est inscrite dans l'objet de configuration d'appareil (DPO).
– A la fin du processus de configuration, la DLE de configuration met Join_Command=1, indiquant l'achèvement réussi du processus de configuration et amenant la DLE à se réinitialiser à l'état configuré. La DLE définit cette réinitialisation comme consistant à réinitialiser d'abord la DLE à son état d'usine, ensuite à effacer le support de l'annonce de la DLE de configuration et ensuite à initialiser la DLE avec les valeurs de réglage stockées dans le DPO.
– La DLE commence alors le fonctionnement de son diagramme d'états, en utilisant la configuration telle qu'initialisée par le DPO. Il convient que cette configuration issue de la DLE de configuration soit appariée au sous-réseau D cible pour faciliter la découverte efficace de sous-réseau D. Par exemple, si le sous-réseau D cible est configuré pour émettre des annonces sur trois voies, le DPO pourrait raisonnablement configurer la DLE pour balayer ces mêmes trois voies.

Le processus de découverte est réussi lorsque la DLE reçoit une annonce issue du sous-réseau D cible. Cette annonce contient une configuration de DLE comprimée, mais

pleinement fonctionnelle, qui peut être utilisée pour la communication avec le gestionnaire de système par l'intermédiaire du routeur qui a émis l'annonce.

Si le processus de rattachement temporise, la DLE est réinitialisée à l'état configuré. La configuration dérivée de l'annonce est effacée, la configuration de DPO est rétablie, et la DLE reprend sa recherche d'un sous-réseau D cible.

Si le processus de rattachement est réussi, la configuration de rattachement de la DLE persiste jusqu'à ce qu'elle soit explicitement mise à jour par le gestionnaire de système. Ainsi, à la fin du processus de rattachement, la DLE maintient habituellement une connexion provisoire avec le gestionnaire de système par l'intermédiaire du routeur d'annonce.

Après un rattachement réussi, la DLE recherche un jeu de routeurs candidats qui peuvent être utilisés pour la communication. Après une période de temps configurable, prenant la valeur par défaut de 60 secondes, la DLE rapporte cette liste de candidats au gestionnaire de système. Ces informations sont utilisées par le système pour remplacer la connexion provisoire par une configuration plus permanente et plus résiliente de la DLE.

Si le taux de rapports prévu de DLE est si bas que le temps de synchronisation exigé pour le saut de voie discrétisé ne peut pas être maintenu et si le sous-réseau D local fournit des intervalles de saut de voie lent, alors le DLE peut changer pour utiliser le saut de voie lent pour la plupart des communications rares.

NOTE Le processus de rattachement, prévu pour une utilisation rare et acyclique de toute DLE donnée, utilise le saut de voie discrétisé; le saut de voie lent n'est pas pris en charge pendant le processus de rattachement.

9.1.14.2 Etats de DLE

Lorsqu'un appareil est manufacturé, la DLE est dans son état par défaut. Les attributs dans l'état par défaut sont définis par la présente norme.

Dans l'état par défaut, la DLE doit rechercher périodiquement par balayage un sous-réseau D de configuration, en utilisant une procédure de recherche définie par la norme. Lorsque la DLE reçoit une ou plusieurs annonces issues d'une DLE de configuration dans un sous-réseau D de configuration, la DLE doit utiliser les informations contenues dans l'une des annonces pour établir une supertrame avec les liaisons qui peuvent être utilisées pour communiquer la DLE de configuration sélectionnée. La DLE informe alors le DPO (objet de configuration d'appareil) qu'une association D a été établie, et commute la DLE vers son état de configuration.

Dans l'état de configuration, la DLE met fin à la procédure de recherche définie par la norme. Plutôt, la DLE actionne son diagramme d'états selon une supertrame avec des liaisons qui ont été fournies par l'annonce de la DLE de configuration. Pendant le processus de configuration, la DLE fournit des services de communication aux couches supérieures en actionnant son diagramme d'états selon la supertrame et les liaisons annoncées, si bien que les APDU de configuration peuvent être passées entre le DPO et la DLE de configuration par l'intermédiaire de la DLE qui est en train d'être configurée.

Si le processus de configuration temporise, la DLE retourne à son état par défaut et reprend sa procédure de recherche par défaut d'un sous-réseau D de configuration.

Si le processus de configuration est réussi, le DPO fournit à la DLE un jeu d'attributs, y compris les informations relatives au sous-réseau D, les supertrames, et les liaisons, que la DLE peut utiliser pour rechercher le(s) sous-réseau(x) D cible(s). La DLE commute alors de l'état "provisioning" vers l'état "provisioned", et actionne son diagramme d'états tel que configuré dans les supertrames et les liaisons qui avaient été fournies par le DPO.

La commutation de l'état "provisioning" vers l'état "provisioned" est déclenchée lorsque la DLE de configuration met Join_Command=1 (Tableau 10). Lorsque cela se produit, la DLE est

réinitialisée à son état "provisioned" et actionne alors son diagramme d'états tel que configuré afin de rechercher un sous-réseau D cible.

En général, une réinitialisation de DLE vers l'état "provisioned" est accomplie en deux étapes. D'abord, la DLE est réinitialisée à son état par défaut. Puis, l'information issue de l'attribut Target_DL_Config du DPO est appliquée à la DLE. Cela fournit un jeu d'attributs, y compris l'information relative au sous-réseau D, les supertrames, et les liaisons, que la DLE utilise pour rechercher le réseau de cible et les sous-réseaux D correspondants D. Voir 13.8.

Si la DLE est configurée par un mécanisme hors bande, l'état "provisioning" peut être contourné et dans ce cas, la DLE passe directement de l'état "unprovisioned"(non configuré) à l'état "provisioned".

Le DPO retient une copie des informations qui avaient été utilisées pour configurer la DLE, fournissant des moyens de réinitialiser la DLE à son état "provisioned" en mettant la DLE dans son état par défaut et en ajoutant ensuite les attributs configurés issus du DPO.

Une DLE dans l'état "provisioned' actionne son diagramme d'états tel que configuré dans ses supertrames et liaisons configurées. Il convient que les supertrames et liaisons configurées soient mises en concordance avec les caractéristiques de fonctionnement du(des) sous-réseau(x) D cible(s), de sorte qu'un réseau-cible soit efficacement découvert lorsque la DLE et un sous-réseau D cible sont placés à proximité l'un de l'autre. Le résultat est que la DLE reçoit au moins une annonce issue d'au moins une DLE proxy qui participe à l'un de ces sous-réseaux D cibles. La DLE utilise les informations acheminées par l'une des annonces reçues pour établir une supertrame avec les liaisons qui peuvent être utilisées pour communiquer avec la DLE proxy émettrice. La DLE qui tente de rejoindre le sous-réseau D informe alors le DMAP qu'une association D a été établie vers un proxy voisin, et que la DLE commute alors de l'état "provisioned" vers l'état "joining" (rattachement).

Lorsque la DLE entre dans l'état "joining", elle maintient sa configuration provenant de l'état "provisioned" (voir 13.8), et ajoute le supertrame, les liaisons et d'autres attributs définis ou sous-entendus par l'annonce. La DLE fournit alors des services de communication aux couches supérieures au cours du processus de rattachement, en actionnant son diagramme d'états selon la supertrame et les liaisons annoncées avec le résultat que les APDU de rattachement sont passées entre le DMAP et la DLE proxy par l'intermédiaire de la DLE qui est en train d'être configurée.

Si le processus de rattachement du DMAP temporise, la DLE se réinitialise à son état "provisioned" et recommence à rechercher par balayage un sous-réseau D en utilisant des supertrames et liaisons configurées.

Si le processus de rattachement du DMAP est réussi, les supertrames et les liaisons annoncés continuent d'être utilisées temporairement pour la communication avec le gestionnaire de système. Le DPO conserve les informations nécessaires pour réinitialiser la DLE vers l'état "provisioned" en cas de réinitialisation de DLE.

Lorsque la DLE est d'abord placée dans son état rattaché, elle a une seule connexion au sous-réseau D par l'intermédiaire de la DLE proxy voisine, utilisant la supertrame et les liaisons définies dans l'annonce. Cette connexion unique, sélectionnée implicitement lorsque la DLE a sélectionné la DLE proxy, manque de diversité de chemin, et peut être suboptimale pour un nombre quelconque de raisons. Par conséquent, le gestionnaire de système doit fournir des instructions pour que la DLE recherche plusieurs voisins et rapporte le résultat lorsque la recherche est achevée. Des instructions simples peuvent être fournies par l'intermédiaire de la même annonce qui avait établi la connexion initiale à la DLE proxy, ou des instructions de recherche plus élaborées peuvent être fournies par le gestionnaire de système immédiatement après que la DLE a rejoint le sous-réseau D. Dans un cas comme dans l'autre, la DLE fournit au gestionnaire de système une liste de voisins candidats peu après qu'elle a rejoint le sous-réseau D, 60 s étant le temps de rapport par défaut tel que commandé par l'attribut dlmo.DiscoveryAlert. Le gestionnaire de système analyse cette liste

de voisins candidats, fournit alors à la DLE une configuration mise à jour qui inclut la diversité de chemin (maille) et fournit généralement une connexion de sous-réseau D plus optimale.

Pour prendre en charge le traitement de sécurité, le DMAP a besoin de l'adresse EUI64Address de la DLE d'annonce pendant le processus tant de configuration que de rattachement. Sachant que la DPDU Data d'annonce fournit seulement l'adresse DL16Address du routeur d'annonce, la DLE doit acquérir l'adresse EUI64Address du voisin lorsque la communication commence. L'adresse EUI64Address du voisin doit être acquise en utilisant une liaison d'émission pour interroger le voisin à l'aide d'une DPDU Data avec une charge utile vide, en mettant à 1 le bit d'adresse EUI64Address de la demande de DHDR (bit 5, Tableau 118), faisant retourner l'adresse EUI64Address du voisin dans la DPDU ACK/NAK.

9.1.14.3 Informations consolidées de configuration de DL

Le DPO maintient l'attribut Target_DL_Config qui inclut les valeurs de réglage pour les divers attributs dans la DLE. Cet OctetString est fourni à la DLE à la fin du processus de configuration, et est conservé par le DPO au cas où il aurait besoin de réinitialiser la DLE à son état "provisioned".

Le DL_Config_Info OctetString contient un ensemble d'attributs que le DPO fournit à la DLE afin d'établir l'état "provisioned". Chaque attribut est exprimé comme un tuple comprenant le numéro d'attribut, suivi d'un OctetString qui contient la nouvelle valeur d'attribut. La structure de DL_Config_Info est montrée dans le Tableau 102.

Tableau 102 – Structure de DL_Config_Info

Octets	Bits							
	7	6	5	4	3	2	1	0
1 octet	N (nombre d'attributs)							
1 octet	$AttributeNumber_1$ (Unsigned8)							
–	$NewAttribute_1$ (OctetString)							
...	...							
1 octet	$AttributeNumber_N$ (Unsigned8)							
–	$NewAttribute_N$ (OctetString)							

Plusieurs des attributs dans DL_Config_Info sont des OctetString indexés. Dans ces cas, $NewAttribute_X$ est une nouvelle entrée de rangée. Par convention de DL, chaque entrée de rangée dans un attribut OctetString indexé inclut l'indice de rangée comme étant son premier champ.

DL_Config_Info peut être utilisé pour configurer n'importe quel attribut de lecture/écriture dans la DLE. A tout le moins, DL_Config_Info doit configurer:

- AdvFilter, qui fournit un filtre de sorte que la DLE puisse sélectionner les supertrames qui sont d'intérêt.
- Au moins une supertrame, et au moins une liaison, qui peuvent être utilisées par la DLE pour rechercher des annonces.

Des modèles d'intervalle de temps utilisés pour rechercher le sous-réseau D, s'ils sont différents des modèles d'intervalle de temps par défaut, seront fournis au DLE au cours du processus de configuration.

DL_Config_Info ne doit pas être utilisé excepté par l'intermédiaire du DPO.

Les attributs configurés de la DLE doivent être conservés par le DPO afin que la DLE puisse être réinitialisé à son état "provisioned".

Le fonctionnement de supertrame peut être retardé ou désactivé en positionnant le champ IdleTimer dans la supertrame. Par exemple, deux supertrames peuvent être configurées dans une DLE, la supertrame numéro1 recherchant agressivement le sous-réseau D alors que la supertrame numéro2 effectue une recherche sur un facteur d'utilisation faible. L'IdleTimer de la supertrame numéro1 pourrait conduire cette supertrame à temporiser quelques minutes après que la DLE a été configurée. En variante, une supertrame pourrait être configurée avec un IdleTimer afin qu'elle soit inactive jusqu'à un certain temps dans le futur. Si la DLE est réinitialisée à l'état "provisioned", les temporisateurs d'inactivité de la supertrame doivent également être réinitialisés à l'état initialement configuré.

Une DLE dans l'état "provisioned" doit actionner son horloge de DL et incrémenter le temps TAI. Cela permet à la DLE d'actionner ses supertrames telles que configurées pour découvrir des sous-réseaux D candidats. Pendant le processus de rattachement, un temps TAI révisé sera reçu dans des annonces et l'horloge de DL sera réinitialisée en conséquence.

La DLE pourrait perdre complètement son sens du temps pendant qu'elle est dans l'état configuré, par exemple en raison du retrait d'une batterie. La perte complète du sens de temps dans une DLE configurée doit déclencher une réinitialisation de la DLE à son état configuré.

9.1.14.4 Recherche de voisins par balayage dans l'état "unprovisioned"

Une DLE non configurée commence dans l'état par défaut du système. Sa configuration de DLE inclut les cinq modèles de saut de voie par défaut et les trois modèles d'intervalle de temps par défaut. Ses supertrames, liaisons, graphes et chemins sont blancs.

Avant d'essayer de rejoindre le réseau d'installation, la DLE non configurée a besoin d'établir le contact avec une DLE de configuration et être passée à l'état configuré. Cela peut se produire par l'intermédiaire d'un mécanisme hors bande, tel qu'un modem câblé ou une liaison infrarouge.

La norme définit une procédure de recherche de DLE non configurée pour une annonce de sous-réseau D de configuration. La procédure de recherche est en silence radio, n'impliquant pas de sollicitations ou toute autre émission jusqu'à ce qu'une annonce soit reçue en provenance d'un sous-réseau D de configuration.

Une DLE non configurée doit rechercher par balayage des annonces d'un sous-réseau de configuration D sur les voies 4 et 14, correspondant aux voies 15 et 25 de l'IEEE 802.15.4, si le règlement des radiocommunications le permet (voir Figure 59). Dans chacune de ces voies, la DLE non configurée doit rechercher par balayage une annonce à un intervalle fixe d'exactement 0,25 s, 0,5 s, 1 s, 2 s, 4 s, 8 s, 16 s, ou 32 s. Lorsque la DLE est mise sous tension ou physiquement réinitialisée, elle doit réaliser le balayage au plus court intervalle de 0,25 s pendant au moins 10 s, et ensuite elle peut augmenter progressivement l'intervalle selon les besoins pour préserver son alimentation en énergie.

SubnetID=1 est réservé au sous-réseau D de configuration. Par conséquent, seules les annonces avec SubnetID=1 doivent être considérées par une DLE dans l'état non configuré.

9.1.14.5 Recherche par balayage de voisins dans l'état "provisioned"

Une fois qu'une DLE est dans l'état configuré, elle doit utiliser ses supertrames et liaisons configurées pour rechercher par balayage un sous-réseau D d'intérêt. La DLE peut découvrir plusieurs routeurs d'annonce et ensuite en sélectionner un devant être utilisé comme proxy dans le processus de rattachement.

9.1.14.6 Recherche de voisins par balayage après rattachement au sous-réseau D

Une DLE rejoint le sous-réseau D par l'intermédiaire d'un seul voisin qui envoie des annonces. Les contrats initiaux sont établis en fonction d'un chemin qui n'est pas

nécessairement optimal, par l'intermédiaire de la DLE proxy se rattachant. Ayant établi une seule connexion par le processus de rattachement, la DLE doit être configurée par le gestionnaire de système pour rechercher immédiatement des voisins supplémentaires et alternatifs afin de prendre en charge une configuration de maille plus optimisée.

Une DLE dans l'état rattaché peut être configurée par le gestionnaire de système pour rechercher, par balayage passif, des annonces en configurant la DLE avec les liaisons et les supertrames qui activent le récepteur radio de la DLE à des temps programmés. La DLE peut être configurée pour rechercher, par balayage actif, des annonces utilisant des sollicitations.

L'alerte NeighborDiscovery (voir 9.6.2 et 9.4.2.24) fournit un mécanisme pour que la DLE transfère ces informations vers le gestionnaire de système. L'alerte est prévue pour les scénarios suivants:

- Dans la limite de 15 s de son entrée dans l'état rattaché, la DLE doit être configurée avec les supertrames et les liaisons qui recherchent des annonces issues de routeurs qui sont dans la portée, et sélectionner les candidats les plus prometteurs. La configuration peut être accomplie par l'intermédiaire de l'annonce elle-même, avant le rattachement, comme décrit en 9.3.5.2.4.2. En variante, la configuration peut être fournie en configurant des attributs de DLMO après le rattachement. Après une quantité de temps écoulée configurable, telle que définie par l'attribut dlmo.DiscoveryAlert, la DLE doit utiliser l'alerte de découverte de voisins pour envoyer la liste de candidats vers le gestionnaire de système. Une supertrame dédiée à cette recherche initiale peut être configurée pour temporiser automatiquement par l'intermédiaire de son champ IdleTimer.
- Il convient que la DLE soit configurée par le gestionnaire de système pour rechercher continuellement, par balayage continu passif et/ou actif, des annonces sur une base continue. Au fil du temps, cela permet à la DLE d'établir une liste de voisins candidats. Le HRCO peut être configuré pour émettre périodiquement cette table de voisins candidats, avec le diagnostic de voisin, à destination du gestionnaire de système. Le gestionnaire de système peut en variante lire la table de voisins candidats, dlmo.Candidates, sur son propre programme.
- Une DLE doit aussi utiliser l'alerte de découverte de voisins chaque fois qu'un problème de connectivité est rapporté par l'intermédiaire de l'alerte de DL_Connectivity (voir 9.6.1). Cela fournit une image mise à jour de la connectivité de voisinage, permettant au gestionnaire de système de considérer immédiatement des solutions de remplacement au problème rapporté de connectivité de DLE.

9.1.15 Commande de liaison radio et mesure de qualité

9.1.15.1 Généralités

Les informations relatives à la qualité de signal sont accumulées dans la DLE et rapportées par l'intermédiaire du DMAP. A l'appui de ces fonctions de plus haut niveau, la DLE fournit des primitives qui rapportent les informations relatives à la qualité de signal. La DLE fournit aussi les attributs qui permettent au gestionnaire de système de commander les émissions radio.

9.1.15.2 Métrique de performances

De nombreuses métriques de performances peuvent être accumulées par la DLE sur une base voisin par voisin, configurées par le gestionnaire de système et rapportées par l'intermédiaire du DMAP (voir 9.4.3.9).

La DLE peut être configurée par le gestionnaire de système pour accumuler les types suivants de données de performances sur une base voisin par voisin:

- Indicateur d'intensité de signal reçu (RSSI) et indicateur de qualité de signal reçu (RSQI).

NOTE La présente norme définit des pratiques pour rendre compte de RSSI et RSQI pour faciliter la cohérence, de sorte que les caractéristiques de signal soient quelque peu comparables entre des appareils de construction différentes.

- Pour les émissions: comptes des émissions réussies, replis de CCA, erreurs en monodiffusion, et NAK.
- Compte des DPDU Data reçues.
- Diagnostics de corrections d'horloge.

Les diagnostics par voie sont également recueillis et consolidés pour tous les voisins.

RSSI doit être rapporté sous la forme d'un nombre entier signé de 8 bits, reflétant une estimation de l'intensité de signal reçu en dBm. Les rapports de RSSI doivent être biaisés de +64 dBm pour donner une étendue de mesure de -192 dBm à +63 dBm. Par exemple, une valeur rapportée de RSSI de -16 correspond à une intensité de signal reçu de -80 dBm.

La force de signal réelle reçue pour un appareil dépend du bruit de fond du receveur, qui dépend de la construction de l'appareil et des températures d'exploitation, ainsi que de la sensibilité minimale du receveur. Ceux-ci peuvent être pris en compte dans l'appareil de réception, car ils sont assez reproductibles pour la conception d'appareil donnée et la température d'exploitation de l'appareil. Néanmoins, il convient que les concepteurs d'appareil établissent un mapping approximatif des indications disponibles sur la force de signal reçue et la température de l'appareil (lorsqu'elle est connue) à une estimation raisonnable de RSSI afin que les gestionnaires de système aient une base cohérente pour prendre des décisions de routage parmi les DLE.

RSQI doit être rapporté comme étant une évaluation qualitative de la qualité de signal, un nombre plus élevé indiquant un meilleur signal. Une valeur de 1..63 indique un signal médiocre, 64..127 un signal passable, 128..191 un bon signal, et 192..255 un excellent signal. Une valeur égale à zéro indique que le jeu de puces ne prend en charge aucun diagnostic de qualité de signal autre que RSSI.

RSSI est une mesure quantitative, mappée sur des unités physiques. RSQI est une mesure qualitative. Les appareils RF de fabricants différents, ou avec des numéros de pièces différents, ou avec un schéma de principe amélioré ou une rétractation photolithographique, peuvent générer des valeurs de RSQI brutes différentes. Etant donné que la PHY de l'IEEE 802.15.4 ne spécifie aucune mesure commune et méthodologie de rapports pour le matériel sous-jacent, aucune sous-couche logicielle supérieure ne peut la créer; l'information n'est simplement pas présente parmi les différents appareils.

Les métriques de RSQI sont destinées à être particulièrement utiles pour comparer différentes entrées dans dlmo.Candidates, où l'évaluation réside dans la signalisation des origines différentes reçues par un même appareil. Une DLE peut utiliser des techniques innovantes pour rapporter des comparaisons de la qualité de liaison probable de différents voisins candidats. Puisque les variances entre appareils sur l'appareil receveur sont supprimées, les entrées de RSQI dans dlmo.Candidates rapportées à partir d'une DLE donnée peuvent être raisonnablement comparées entre elles, et des distinctions fines peuvent être considérées comme étant sensées. Par exemple, des différences au sein de la gamme de bons signaux peuvent être raisonnablement prises en compte si les métrique de RSQI sont issues de la DLE.

RSQI peut également être comparé à travers des DLE différentes, mais les distinctions fines sont peu susceptibles d'être aussi sensées. Par exemple, la distinction entre une liaison passable et une liaison excellente est susceptible d'être sensée même si elle est rapportée à partir d'appareils dissemblables, mais les distinctions entre différents niveaux de bonnes liaisons n'ont aucune signification normalisée si elles sont rapportées à partir d'appareils différents.

Etant donné que toute valeur RSQI rapportée est une mesure qualitative, la comparaison de telles valeurs doit nécessairement prendre en compte la nature spécifique de la puce RF des

mesures en question. Etant donné l'interprétation spécifique des valeurs rapportées, deux valeurs RSQI différentes de zéro rapportés par des DLE différentes variant d'une quantité de 32 ou plus peuvent être classées "meilleure" ou "pire", la fiabilité du classement augmentant à mesure que les valeurs numériques augmentent.

De façon similaire, des valeurs RSSI rapportées par le même appareil à des temps différents ou pour des correspondances à distance différentes peuvent être de façon fiable comparées, tout comme les valeurs RSSI parmi des appareils qui sont connus (par le fournisseur ou tout autre identification de modèle spécifique à l'appareil) pour fournir des estimation RSSI calibrées. Dans d'autres cas, de telles comparaisons sont au mieux approximatives, quelque peu similaires aux RSQI bien que présumées plus proches de l'approximation de la force réelle du signal reçu par l'appareil rapporteur. En particulier, les ordres de grandeur de magnitude parmi les rapports du même appareil rapporteur sont toujours fiables en termes d'ordre de grandeur et de magnitude de différence approximative.

9.1.15.3 Accumulation et comptes-rendus d'informations de diagnostic

Le gestionnaire de système établit une relation de communication de DL entre une DLE et son voisin en ajoutant une entrée à l'attribut dlmo.Neighbor de la DLE. Chaque entrée indique un niveau du diagnostic devant être collecté, par l'intermédiaire du champ dlmo.Neighbor[ ].DiagLevel. Pour chaque voisin, le diagnostic peut être recueilli à un niveau de ligne de base, ou à un niveau détaillé incluant le diagnostic d'horloge.

Les diagnostics par voie sont accumulés et consolidés pour tous les voisins, dans l'attribut dlmo.ChannelDiag.

Lorsque le champ dlmo.Neighbor[ ].DiagLevel est mis pour un voisin particulier, la DLE doit créer les entrées correspondantes dans l'attribut en lecture seule dlmo.NeighborDiag. Les valeurs de NeighborDiag sont accumulées à partir du temps où l'entrée de dlmo.NeighborDiag est créée.

Trois mécanismes sont fournis pour rapporter les informations de diagnostic contenues dans dlmo.NeighborDiag et dlmo.ChannelDiag:

- L'objet concentrateur de rapports de santé (HRCO), décrit en 6.2.7.7, peut être configuré pour rapporter tout attribut dans la DLE sur une base périodique. Les entrées de dlmo.NeighborDiag et de dlmo.ChannelDiag peuvent être rapportées par l'intermédiaire du mécanisme en question.
- Les informations de diagnostic peuvent être récupérées à tout moment par le gestionnaire de système, en lisant les attributs applicables.
- Les informations de diagnostic peuvent être rapportées par la DLE sur une base d'exception, par l'intermédiaire de l'alerte de DL_Connectivity.

Les diagnostics incluent une combinaison de niveaux, tels que RSSI, et des compteurs, tels qu'un compte d'acquittements.

Des niveaux sont accumulés sous forme de moyennes mobiles exponentielles (EMA). Le niveau est initialisé avec la première valeur de données, après laquelle chaque nouvelle valeur de données est accumulée dans le niveau d'EMA comme suit, où:

$$EmaLevel_{NEW} = EmaLevel_{OLD} + (\alpha / 100) \times (NewData - EmaLevel_{OLD})$$

Le facteur de lissage α est exprimé sous la forme d'un nombre entier dans la plage 0..100, représentant un pourcentage; il est configuré par le gestionnaire de système par l'intermédiaire de l'attribut dlmo.SmoothFactors (voir 9.4.2.25).

Les compteurs dans dlmo.NeighborDiag sont accumulés comme des nombres entiers non signés ExtDLUint, qui sont intérieurement des entiers de 15 bits. Lorsqu'un compteur atteint sa valeur maximale, de 32 767 (0x7FFF), il doit "se bloquer" et continuer de rapporter cette

valeur maximale. Les compteurs doivent être réinitialisés à zéro chaque fois que la rangée est rapportée par l'intermédiaire du HRCO ou récupérée par l'intermédiaire d'une opération "read" (lecture). Le fait de rapporter la valeur par l'intermédiaire de l'alerte de DL_Connectivity ne doit réinitialiser aucun compteur.

9.1.15.4 Silence radio

La DLE peut être configurée pour émettre seulement lorsqu'elle participe activement à un sous-réseau D. Ce comportement est configuré par l'attribut dlmo.RadioSilence, qui désigne une période de temporisation pour la participation à un sous-réseau D, en secondes. Par exemple, si l'attribut dlmo.RadioSilence est mis à la valeur par défaut de 600 s (10 min), la DLE met en silence son émetteur radio 10 min après avoir perdu la communication avec le sous-réseau D. Lorsque toutes les DLE sur un sous-réseau D sont configurées pour le silence radio, il est possible de désactiver entièrement le sous-réseau D, même si certaines DLE ne reçoivent pas une commande explicite de désactiver des communications.

Lorsqu'une mise à jour de temps valide est acceptée par la DLE à partir d'une annonce ou d'un acquittement, la DLE enregistre intérieurement le temps courant comme référence de temps de silence radio. Si la DLE n'accepte pas une autre mise à jour de temps dans la période de temps suivante désignée par dlmo.RadioSilence, la DLE doit devenir silencieuse en ignorant l'ensemble de toutes ses liaisons d'émission configurées, y compris les sollicitations. Dans l'état de silence radio, la DLE continue d'actionner son récepteur radio selon ses liaisons de réception programmées, mais sans émettre d'acquittements en l'absence d'une mise à jour d'horloge.

Par exemple, supposons que la DLE reçoive une mise à jour de temps 01h:02m:03s et que dlmo.RadioSilence est mis à 600 s (10 min). Si la DLE ne reçoit pas d'autre mise à jour de temps au plus tard à 01h:12m:03s, soit 10 min plus tard, elle mettra alors sa radio en silence.

Si dlmo.RadioSilence est configuré comme étant zéro, l'appareil est désactivé.

Le silence radio est la valeur par défaut. La procédure de découverte de sous-réseau D par défaut n'utilise pas de sollicitations, et dlmo.RadioSilence prend la valeur par défaut de 600 s.

La prise en charge de l'attribut dlmo.RadioSilence est exigée pour toutes les DLE.

Le profil de silence radio limite la plage autorisée de l'attribut dlmo.RadioSilence. Le profil de silence radio est rapporté au gestionnaire de système lors du rattachement par l'intermédiaire de dlmo.DeviceCapability. Une DLE avec le profil de silence radio doit rejeter les mises à jour de dlmo.RadioSilence qui sont à l'extérieur de la plage 1 s à 600 s, assurant ainsi qu'une telle DLE n'émettra jamais une DPDU une fois qu'elle a perdu le contact avec le sous-réseau D pendant 600 s.

Le silence radio temporaire peut être accompli avec un autre attribut dlmo.RadioSleep. Lorsque dlmo.RadioSleep est mis à une valeur positive, la DLE traite toutes les liaisons, y compris les liaisons de réception, comme étant inactives pendant le nombre désigné de secondes. Le silence radio temporaire peut être accompli avec un autre attribut dlmo.RadioSleep. Lorsque dlmo.RadioSleep est mis à une valeur positive, la DLE traite toutes les liaisons, y compris les liaisons de réception, comme étant inactives pendant le nombre désigné de secondes. L'activation de dlmo.RadioSleep doit être légèrement retardée pour permettre la prise en charge de l'émission d'un acquittement AL pour la APDU du DMAP qui a fait positionner l'attribut. Lorsque la période de sommeil est terminée, dlmo.RadioSleep est automatiquement réinitialisé à zéro, indiquant que la caractéristique est désactivée.

9.1.15.5 Puissance d'émission radio

La norme fournit au gestionnaire de système un degré de commande sur la puissance d'émission radio, par l'intermédiaire de l'attribut dlmo.RadioTransmitPower.

L'attribut dlmo.RadioTransmitPower est utilisé pour commander le niveau de puissance d'émission radio de la DLE, en dBm EIRP. Il prend comme valeur par défaut le niveau de puissance maximale prise en charge et il est toujours contraint par dlmo.CountryCode (9.1.15.6) aux contraintes de réglementations du lieu d'utilisation. Cette valeur par défaut contrainte est également rapportée au gestionnaire de système au cours du processus de rattachement par l'intermédiaire de dlmo.DeviceCapability.

Lorsque dlmo.RadioTransmitPower est changé par le gestionnaire de système, la DLE ne doit pas émettre à un niveau de puissance de sortie au-dessus de dlmo.RadioTransmitPower.

En outre, une DLE peut de façon autonome étalonner son niveau de puissance de sortie par rapport au niveau minimal exigé pour maintenir une connectivité fiable. Pour permettre cela, la DLE prend en charge le renvoi en écho des informations relatives à la qualité de signal dans les acquittements, de sorte qu'une mise en œuvre peut étalonner la qualité de signal reçu à divers niveaux de puissance. Voir 9.3.5.5.

NOTE Un calcul exact du niveau de sortie réel des DLE depuis les rapports correspondants des RSSI reçues dépend des informations de conception de la DLE correspondante reportant, qui n'est pas connue par la DLE auto-étalonnée. Ces facteurs incluent le bruit de fond et la sensibilité reçue minimale des DLE correspondantes. Ainsi, tout auto-ajustement des niveaux de transmission basés sur les RSSI reçues est au mieux approximatif, en particulier lorsque les sous-systèmes RF des DLE correspondantes ne partagent pas une conception commune (lorsqu'ils proviennent de constructeurs différents, par exemple).

Il est possible pour les DLE de fournir des corrections locales des valeurs RSSI qu'elles rapportent pour rendre compte de l'influence du bruit de fond et de la sensibilité reçue minimale sur leur propre receveur. De telles corrections, non adressées par la norme IEEE 802.15.4, peuvent habituellement être réalisées de façon linéaire et par partie. Les valeurs rapportées résultant sont considérés comme satisfaisant aux exigences RSSI de la présente norme et de l'IEEE 802.15.4, même si elles sont légèrement ajustées à partir des mesures des sous-sections RF des DLE de mise en œuvre (puisque ces mesures réelles ne prennent pas en compte les autres caractéristiques pertinentes du sous-système RF). Voir également 9.1.15.2.

9.1.15.6 Code de pays

La DLE de configuration et/ou le gestionnaire de système peu ou peuvent informer la DLE configurée des considérations de réglementations par l'intermédiaire de son attribut dlmo.CountryCode, Tableau 103, qui est une structure condensée de 16 bits composée d'un code de pays de 10 bits et de six booléens:

- Les bits 0..9 fournissent un code de pays de 10 bits sous la forme d'un nombre entier Unsigned10, utilisant les codes de pays numériques à trois chiffres selon l'ISO 3166-1.
- Les bits 10..15 spécifient une formation à six éléments booléens:
  - Le bit 10 (Index 0) indique si les règles de la FCC s'appliquent. Une DLE doit opérer en conformité avec les règles FCC lorsque l'Index0 (Bit10) se vérifie.
  - Bit 11 (Index 1), ETSI indique si les règles EU (de la norme ETSI) s'appliquent. Une DLE doit opérer en conformité avec les règles EU lorsque l'Index1 (Bit11) se vérifie.
  - Bit 12 (Index 2), LP, indique si une limite de 10 dBm EIRP s'applique. Une DLE doit limiter ses émissions à ≤ 10 dBm lorsque l'Index2 (Bit12) se vérifie.
  - Bit 13 (Index3), LBT, indique si la DLE doit opérer sous des règles d'adaptabilité, en utilisant LBT, pour détecter la voie lors de l'initiation d'une transaction et cesser l'utilisation du connecteur si l'activité est détectée: ne se vérifie pas=non adaptatif, se vérifie= adaptatif.
  - Bit 14 (Index14), FHSS, indique si la DLE doit opérer sous des règles d'étalement de spectre à saut de fréquence: ne se vérifie pas=règles-non-FHSS, se vérifie= règles-FHSS.
  - Le bit 15 (Index15), Locked, indique si la valeur de cet attribut est fixe lorsque la DLE est opérationnelle. Une fois que ce bit "collant" est mis, toute tentative

ultérieure de modifier l'attribut dlmo.CountryCode doit être rejetée, sauf lorsqu'elle la DLE est réinitialisée à ses valeurs de réglage d'usine par défaut au cours de la (re)configuration.

Tableau 103 – CountryCode

Numéro d'octet	Bits							
	7	6	5	4	3	2	1	0
1	Mode						ISO 3166-1 CountryCode bits 9..8	
	Locked	FHSS	LBT	PCLF	ETSI	FCC		
2	ISO 3166-1 CountryCode bits 7..0							

Lorsque les Bit11 et Bit13 (ETSI et LBT) se vérifient tous les deux, chaque transaction D doit commencer avec un intervalle d'observation LBT d'au moins 20 us, en utilisant le mode 1 de CCA, prenant ainsi en charge les modes décrits en Annexe V, 3) et Annexe V, 6).

NOTE 1 Le mode 3 de CCA est également acceptable sous ETSI EN 300 328 v1.8.1 lorsqu'il est mis en œuvre en tant que AND du mode 1 de CCA et du mode 2 de CCA mais n'est pas acceptable lorsqu'il est mis en œuvre en tant que OR du mode 1 de CCA et du mode 2 de CCA. Voir 9.1.9.4.3.

Lorsque les Bit11, Bit13 et Bit14 (ETSI, LBT et FHSS) se vérifient tous, l'exploitation change momentanément vers les règles non adaptatives de l'ETSI EN 300 328 v1.8.1 pendant l'envoi d'une DPDU ACK/NAK (en tant que SCS (Short Control Signaling)) au sein d'une transaction et pour le Tx-gap-time d'une non-émission mandatée-EN immédiat suivant, prenant ainsi en charge le mode décrit en Annexe V, 6).

Le Bit15 (Locked) prend en charge l'exploitation d'appareil (lorsqu'il se vérifie) dans des régimes de réglementation qui interdisent l'aptitude à reconfigurer un appareil de telle manière qu'il viole les contraintes de réglementation, tout en prenant encore en charge les appareils (lorsqu'il ne se vérifie pas) sur des plateformes mobiles telles que les navires et les trains qui peuvent franchir des frontières juridictionnelles de réglementation et tout en permettant toujours (via l'exception de reconfiguration) la réparation ou la remise à neuf d'appareils avec une revente ou une réutilisation ultérieure dans des marchés où d'autres réglementations s'appliquent.

Lorsque aucun pays spécifique d'utilisation prévue n'a été identifié, la valeur par défaut pour dlmo.CountryCode doit être 0x3C00, indiquant qu'il convient qu'un appareil dans l'état par défaut se conforme aux règles de l'US FCC, aux règles de l'UE, à la limite < 10 dBm p.i.r.e, et soit classé comme appareil adaptatif non-FHSS. Voir 5.2.5 et Annexe V.

NOTE 2 Cette valeur par défaut assure que l'équipement, avant d'être configuré, satisfait aux exigences de réglementation de la plupart des régions dans lesquelles il pourrait être déployé, et en particulier comme de telles règles s'appliqueraient à la configuration par défaut à trois voies utilisée pour la configuration par liaison radio dès le déballage. Une telle contrainte permet à l'appareil de participer à la configuration de courte portée sur un support sans fil de Type A, point auquel l'attribut dlmo.CountryCode serait changé pour refléter le régime de réglementation prévu qui s'applique au lieu initial (et habituellement unique) de déploiement de l'appareil.

9.1.16 Rôles et options de DLE

La DL spécifiée par la présente norme est conçue dans le but général de contraindre la gamme des options de construction pour un appareil conforme, tout en permettant des solutions de système flexibles et innovantes.

Le cadre de DL n'exige pas que toutes les DLE soient équivalentes. Par exemple, certains routeurs, conçus comme des appareils d'infrastructure dédiés, pourraient avoir une source continue d'énergie, des processeurs puissants, et une capacité mémoire essentiellement illimitée. Par contraste, certains instruments de terrain peuvent avoir des batteries de faible capacité et peuvent manquer de capacité de routage.

Ces distinctions parmi les DLE sont couvertes de trois façons générales dans la présente norme:

- capacité mémoire;
- capacités de DLE; et
- rôles de DLE.

Chaque DLE a une quantité limitée de mémoire qui est disponible pour des opérations de DL, et le gestionnaire de système a besoin de la connaissance de ces limitations afin de configurer la DLE et équilibrer le fonctionnement du sous-réseau D. La mémoire DL de la DLE n'est pas rapportée sous forme d'un seul bloc, mais plutôt comme des capacités de mémoire spécifiques pour des besoins spécifiques. Par exemple, chaque attribut OctetString indexé prend en charge un nombre limité d'entrées, avec la capacité disponible au gestionnaire de système comme métadonnées. De même, la capacité de tampon pour la transmission de DPDU Data est rapportée par la DLE au démarrage.

Certaines capacités de DLE sont également rapportées au démarrage. Par exemple, la DLE rapporte la stabilité de sa propre horloge, ainsi qu'une liste de voies radio qu'elle peut prendre légalement en charge. Les capacités de DLE rapportées avec la demande de rattachement sont énumérées en 9.4.2.23.

Les rôles de DLE décrivent les capacités générales d'une configuration donnée de DLE. Par exemple, une DLE peut être capable de routage ou pas. Les distinctions de ce type ont diverses implications dans toute la DLE, en termes de capacité mémoire minimale, de capacités de DLE, et de prise en charge de diverses caractéristiques. La DLE rapporte simplement quels rôles elle prend en charge, et le gestionnaire de système est alors responsable d'en faire un mapping dans un portefeuille de capacités de DLE. Des mappings normalisés entre les rôles et les capacités minimales sont fournis en Annexe B.

9.1.17 Considérations relatives à l'énergie des DLE

Les appareils ont différents niveaux d'énergie disponible. Un appareil peut avoir une source d'énergie continue. Un autre appareil peut avoir une grande batterie, mais a besoin de la majeure partie de cette capacité d'énergie pour faire fonctionner un capteur. Un autre appareil encore peut utiliser la récupération d'énergie comme sa source d'énergie primaire. Les compositions chimiques différentes des batteries ont des caractéristiques différentes, une chimie donnée de batterie peut fournir des performances différentes selon le fournisseur, et une capacité de batterie peut varier selon des facteurs environnementaux. Les nouvelles technologies de batterie sont susceptibles d'émerger avec des caractéristiques de performance actuellement inconnues. Une application pourrait avoir besoin d'une durée de vie de batterie de 20 ans, alors qu'une application différente pourrait tolérer une durée de vie de six mois.

La DLE peut être configurée par le gestionnaire de système pour consommer différentes quantités d'énergie. La DLE consomme de l'énergie de deux manières générales:

- La DLE consomme de l'énergie en fournissant le service sans fil à ses propres applications. Lorsqu'une DLE établit un contrat pour émettre des données toutes les 5 s, la DLE consomme une quantité correspondante d'énergie.
- La DLE consomme de l'énergie en agissant comme un routeur pour le compte de DLE voisines. Une DLE peut être configurée pour émettre des annonces toutes les 10 s. Une DLE peut être configurée pour actionner son récepteur presque sans interruption, se mettant à l'écoute pour déceler des sollicitations. Le sous-réseau D peut être configuré de sorte qu'une DLE transmette jusqu'à 100 DSDU par minute. Tous ces scénarios consomment de l'énergie.

La DLE rapporte un sens général de sa capacité de prendre en charge des opérations de routage de DL dans certains champs des attributs dlmo.EnergyDesign. Cet attribut est rapporté par l'intermédiaire de l'attribut dlmo.DeviceCapability.

L'attribut dlmo.EnergyDesign indique la capacité désignée en énergie de l'appareil pour traiter les opérations de DL. Cet attribut est constant au cours de la vie de l'appareil et reflète la conception de l'appareil, pas son état courant. Il convient qu'un gestionnaire de système configure une DLE dans ces limitations énoncées relatives à l'énergie:

– EnergyLife indique la vie de l'énergie de l'appareil par conception. Une valeur positive fournit la vie d'énergie en jours; une valeur négative fournit la grandeur de la vie d'énergie en heures. Une valeur de 0x7FFF indique une source d'énergie continue et l'absence de limitations contraignantes relatives à l'énergie d'appareil. D'autres champs EnergyDesign décrivant la capacité en énergie de la DLE sont basés sur cette vie d'énergie ciblée. La configuration de la DLE au-delà de ces capacités en énergie énoncées est susceptible de réduire la vie d'énergie de l'appareil.
– ListenRate indique la capacité en énergie de la DLE en moyenne, en secondes par heure, pour actionner son récepteur radio. ListenRate inclut le temps pour recevoir des DPDU Data pour les propres contrats d'application de la DLE, plus des DPDU Data transmises par la DLE pour le compte d'autres DLE.
– TransmitRate indique la capacité en énergie de la DLE, en DPDU Data par minute, d'émettre des DPDU Data pour son propre compte et de transmettre des DPDU Data pour le compte de ses voisins.
– AdvRate indique la capacité en énergie de la DLE, en DPDU Data par minute, d'émettre des DPDU Data d'annonce (ou de sollicitation) dédiées.

La constante EnergyDesign ne reflète pas l'état changeant de la source d'énergie d'un appareil. L'attribut dynamique en lecture seule dlmo.EnergyLeft peut être utilisé pour rapporter la capacité d'énergie restante de l'appareil. Une valeur positive indique la vie restante en jours et une valeur négative indique la vie restante en heures. Une valeur de 0x7FFF indique que la caractéristique n'est pas prise en charge. L'attribut dlmo.EnergyLeft est rapporté à la suite du démarrage par l'intermédiaire de dlmo.DeviceCapability, et peut également être rapporté périodiquement par l'intermédiaire du HRCO.

9.2 DDSAP

9.2.1 Généralités

Le DDSAP prend en charge l'acheminent par sauts multiples d'une DSDU (par exemple, et d'une NPDU) entre des DLE dans un sous-réseau D.

DD-DATA.request prend une DSDU issue de la NLE, ajoute un en-tête de DPDU Data comme préfixe, et l'ajoute à la file d'attente de messages. DD-DATA.confirm rapporte ultérieurement si la DSDU a été acheminée avec succès vers une DLE voisine dans le sous-réseau D.

DD-DATA.indication indique la réception d'une DPDU Data qui a atteint sa destination finale au sein du sous-réseau D, et passe sa DSDU à la NLE.

Toutes les interfaces entre la DLE et les entités de couche ou entités de gestion adjacentes sont des interfaces internes au sein de l'appareil, et sont donc inobservables. Par conséquent, elles sont strictement virtuelles et ne sont pas soumises à une normalisation.

9.2.2 DD-Data.request

DD-DATA.request est une primitive qui accepte une DSDU provenant de la NL, sélectionne le chemin à travers le sous-réseau D, et place une DPDU Data correspondante dans la file d'attente de messages de la DLE.

La sémantique de la primitive DD-DATA.request est comme suit:

DD-DATA.request (
SrcAddr,
DestAddr,
Priority,
DE,
ECN,
LH,
ContractID,
DSDUSize,
DSDU,
DSDUHandle)

Le Tableau 104 décrit les paramètres pour DD-DATA.request.

Tableau 104 – Paramètres DD-Data.request

Nom du paramètre	Type du paramètre
SrcAddr (DL source address)	Type: Adresse DL16Address ou adresse EUI64Address
DestAddr (adresse de destination DL)	Type: Adresse DL16Address ou adresse EUI64Address
Priorité (priorité de la charge utile)	Type: Unsigned4
Discard Eligible (éligible au rejet)	Type: Unsigned1
ECN (notification d'encombrement explicite)	Type: Unsigned2
LH (dernier saut, NL)	Type: Unsigned1
ContractID (ContractID de la charge utile)	Type: Unsigned16 ou null
DSDUSize (taille de la charge utile)	Type: Unsigned8
DSDU (nombre d'octets par DSDUSize)	Type: Octets
DSDUHandle (identifie de manière unique chaque invocation de cette primitive)	Type: Extrait

Les paramètres de DD-DATA.request comprennent:

- SrcAddr est l'adresse source de la NSDU. Il s'agit normalement du pseudonyme de l'adresse DL16Address de l'adresse IPv6Address source de la NSDU, excepté lorsqu'il s'agit de l'adresse EUI64Address d'une DLE non rattachée. Le paramètre Subnet ID est implicite, basé sur dlmo.SubnetID.
- DestAddr est l'adresse de destination de la NSDU. Il s'agit normalement du pseudonyme de l'adresse DL16Address de l'adresse IPv6Address de destination de la NSDU, excepté lorsqu'il s'agit de l'adresse EUI64Address d'une DLE non rattachée. Le paramètre Subnet ID est implicite, basé sur dlmo.SubnetID.
- Le paramètre Priority est copié dans le sous-en-tête DROUT et indique la priorité de la DPDU Data dans des files d'attente de messages de DLE.
- Le paramètre DE est copié dans le sous-en-tête DADDR. DE=1 indique que la DSDU est éligible à être rejetée d'une file d'attente de messages en faveur d'une DPDU Data entrante avec DE=0, et de priorité égale ou plus haute.

NOTE L'expression "est éligible" ne signifie pas "est obligatoire". Un tel rejet est une option de mise en œuvre.

- Le paramètre ECN est copié dans le sous-en-tête DADDR. Voir 9.1.9.4.5 pour un débat relatif à l'ECN.
- Le paramètre LH est copié dans le sous-en-tête DADDR. Une valeur de 1 indique que la DSDU est entrée dans le sous-réseau D par un routeur dorsal, et donc elle ne doit pas sortir de la DL par un routeur dorsal pour éviter les chemins circulaires à la NL. Cela permet à la NL d'élider le champ limite de saut de l'IPv6. Logiquement, le paramètre LH est transporté par la DL pour le compte de la NL, et LH ne doit pas être changé par la DL.

- ContractID peut être utilisé par la DLE dans la sélection de chemin (voir le débat en 9.1.6.5).
- DSDUSize indique le nombre d'octets contenus dans la charge utile de DPDU Data.
- Le paramètre DSDU est le jeu d'octets formant la charge utile. Il peut être mis en œuvre comme pointeur vers la mémoire qui est partagée parmi les couches.
- Le paramètre DSDUHandle est une abstraction qui connecte chaque invocation de DD-DATA.request avec le rappel ultérieur par DD-DATA.confirm.

9.2.3 DD-Data.confirm

DD-DATA.confirm est une primitive qui rapporte les résultats d'une demande d'émettre une DSDU qui a été précédemment placée dans la file d'attente de messages de la DLE par DD-DATA.request.

Le Tableau 105 décrit les paramètres pour DD-DATA.confirm.

Tableau 105 – Paramètres DD-Data.confirm

Nom du paramètre	Type du paramètre
DSDUHandle (identificateur pour la charge utile)	Type: Extrait
Status (voir le Tableau 106)	Type: Unsigned

Le Tableau 106 spécifie la valeur pour le paramètre "status".

Tableau 106 – Jeu de valeurs pour le paramètre "status"

Valeur	Description
SUCCESS	L'opération a réussi
FAILURE	L'opération a échoué, l'opération a expiré

NOTE Le traitement des erreurs entre la DLE et la NLE contiguë est une question d'appareil interne, qui n'est visible à travers aucune interface observable et n'est donc pas normalisé.

9.2.4 DD-Data.indication

DD-DATA.indication est une primitive virtuelle qui indique la réception d'une DSDU. Une DPDU Data ne déclenche pas d'indication de données tant qu'elle n'a pas atteint sa destination dans le sous-réseau D.

La sémantique de la primitive DD-DATA.indication est comme suit:

```
DD-DATA.indication(
                SrcAddr,
                DestAddr,
                Priority,
                DE,
                ECN,
                LH,
                DSDUSize,
                DSDU)
```

Le Tableau 107 décrit les paramètres pour DD-DATA.indication.

Tableau 107 – Paramètres DD-Data.indication

Nom du paramètre	Type du paramètre
SrcAddr	Type: Adresse DL16Address ou adresse EUI64Address
DestAddr	Type: Adresse DL16Address ou adresse EUI64Address
Priorité (priorité de la charge utile)	Type: Unsigned4
Discard Eligible (éligible au rejet)	Type: Unsigned1
ECN (notification d'encombrement explicite)	Type: Unsigned2
LH (dernier saut, NL)	Type: Unsigned1
DSDUSize (taille de la charge utile)	Type: Unsigned8
DSDU (nombre d'octets par DSDUSize)	Type: Octets

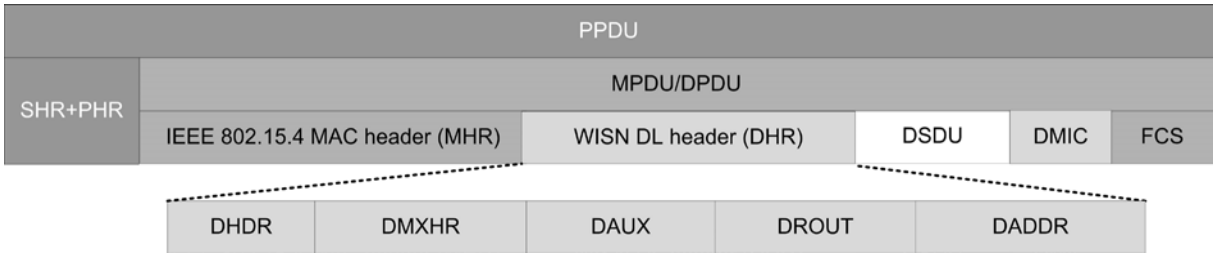
Les paramètres de DD-DATA.indication comprennent:

- SrcAddr est l'adresse source de la NSDU. Il s'agit normalement du pseudonyme de l'adresse DL16Address de l'adresse IPv6Address source de la NSDU, excepté lorsqu'il s'agit de l'adresse EUI64Address d'une DLE non rattachée. Le paramètre D-subnet ID est implicite, basé sur dlmo.SubnetID.
- DestAddr est l'adresse de destination de la NSDU. Il s'agit normalement du pseudonyme de l'adresse DL16Address de l'adresse IPv6Address de destination de la NSDU, excepté lorsqu'il s'agit de l'adresse EUI64Address d'une DLE non rattachée. Le paramètre D-subnet ID est implicite, basé sur dlmo.SubnetID.
- Le paramètre Priority est inclus dans le sous-en-tête DROUT et peut être utilisé par la NL pour routage ultérieur. ContractID, si exigé par la NL, n'est pas transporté dans l'en-tête de la DPDU Data.
- Le paramètre DE fournit la valeur du bit DE copiée à partir du sous-en-tête DADDR entrant.
- Le paramètre ECN fournit la valeur du bit ECN copiée à partir du sous-en-tête DADDR entrant et il correspond au bit ECN décrit dans l'IETF RFC 3168. Voir 9.1.9.4.5 pour un débat relatif à l'ECN.
- Le paramètre LH fournit la valeur du bit LH copiée à partir du sous-en-tête DADDR entrant.
- DSDUSize indique le nombre d'octets contenus dans la charge utile de DPDU Data.
- Le paramètre DSDU est le jeu d'octets formant la charge utile. Il peut être mis en œuvre comme pointeur vers la mémoire qui est partagée parmi les couches.

9.3 Data DPDU et DPDU ACK/NAK

9.3.1 Généralités

La structure des DPDU utilisées par la présente norme est montrée à la Figure 88.



Légende

Anglais	Français
IEE 802.15.4 MAC header (MHR)	En-tête (MHR) MAC IEE 802.15.4
WISN DL header (DHR)	En-tête WISN DL (DHR)

Figure 88 – Structure de PhPDU et de DPDU

La DPDU reflète la structure multi-couche de la PDU décrite en 9.1.4, y compris:

- En-tête MAC (MAC header (MHR)): Le MHR est une structure de données modelée sur celle de l'IEEE 802.15.4, telle que spécifiée en 9.1.4 et 9.1.5, qui inclut des informations relatives aux formats de trames et des informations d'adressage des sous-réseaux D. La FCS à la fin de la DPDU est logiquement associé au MHR.
- En-tête de DPDU (DPDU header (DHR): Les informations d'en-tête de DL suivent le MHR. Les sous-en-têtes au sein du DHR comprennent:
  - En-tête de DHR (DHR header (DHDR)): DHDR inclut les valeurs de réglage pour diverses sélections de DLE et un numéro de version.
  - Sous-en-tête d'extension de DHR MAC (DMXHR): Des champs complémentaires, non indiqués par l'IEEE 802.15.4, qui sont nécessaires pour envoyer une DPDU Data vers un voisin immédiat et pour recevoir une DPDU ACK/NAK immédiate. Le DMXHR comprend des informations relatives à l'intégrité cryptographique et aux mesures de confidentialité qui s'appliquent à la DPDU. Le DMIC suivant la DSDU est logiquement associé au DMXHR.
  - Sous-en-tête auxiliaire de DHR (DAUX): Certaines DPDU incluent des informations auxiliaires pour faciliter la découverte de voisins, la propagation de temps, l'échange d'informations, et l'échange de commandes parmi des voisins immédiats. Le sous-en-tête DAUX est fréquemment absent. Un champ DAUX non nul doit être inclus dans des DPDU d'annonce ou de sollicitation dédiées ou, en variante, il peut être intégré dans des DPDU Data n'ayant aucun rapport.
  - Sous-en-tête de routage de DHR (DROUT): Le champ DROUT contient des informations nécessaires pour acheminer la charge utile DPDU à travers le sous-réseau D. Un champ DROUT non nul doit inclure une classe de priorités de DPDU Data et une limite de transmission, plus soit un GraphID, soit un routage de source.
  - Sous-en-tête d'adresse de DHR (DADDR): Le champ DADDR contient la source et la destination des adresses D des points d'extrémité au sein d'un sous-réseau D, avec les champs NL ECN, DE et LH qui sont tous acheminés et visibles par la DL.
- DSDU: La charge utile de la couche supérieure de DPDU est une NPDU unique 6LoWPAN, tel que défini dans l'Article 10, qui est passé à la DLE en tant que DSDU. La charge utile est acheminée de manière transparente au sein du sous-réseau D.
- DMIC: La DMIC près de la fin de la DPDU est logiquement associé au DMXHR. La DMIC est un code d'intégrité fort du point de vue cryptographique qui permet de déterminer que la DPDU reçue
  - a été générée par un appareil qui partage la clé de cryptage pertinente, et
  - n'a pas été altérée avant la réception.

NOTE Dans certains cas la clé de cryptage DMIC pertinente est statique et publiée, permettant la falsification par un attaquant non informé.

- FCS: La FCS à la fin de la DPDU est logiquement associé au MHR. La FCS est un code d'intégrité trivialement falsifiable qui permet la détection d'erreurs DPDU induites par PhL.

Certaines classes de DPDU qui sont générées par la DLE, telles que les annonces et sollicitations dédiées, ont des champs DROUT, DADDR et DSDU nuls.

9.3.2 Ordonnancement d'octets et de bits

9.3.2.1 Généralités

Excepté dans la DL, la présente norme utilise des conventions d'émission et de documentation d'octet de poids fort en premier lieu (MSB ou "big-endian" c'est-à-dire gros-boutiste), suivant le précédent établi par l'ISO/IEC, l'IETF, et beaucoup d'autres. A savoir:

- pour des valeurs à plusieurs octets, l'octet de poids fort est émis le premier; et
- la documentation des octets montre le bit 7 à gauche et le bit 0 à droite.

Cependant, l'IEEE 802.15.4 utilise les conventions de l'octet de poids faible en premier lieu (LSB ou "little-endian", petit-boutiste). A savoir:

- pour des valeurs à plusieurs octets, l'octet de poids faible est émis le premier.

NOTE La spécification de l'IEEE n'est pas entièrement cohérente sur ce point: les sous-en-tête de sécurité de l'IEEE 802.15.4:2011 utilisent des conventions d'émission et de documentation de MSB.

L'ordre d'émission des bits au sein d'un octet est traité à la PhL. Au sein de la DL le débat relatif au MSB et au LSB est limité à l'ordonnancement des octets.

En conséquence, la DL est inévitablement "mixed-endian" (boutiste mélangée), certaines sections utilisant l'ordonnancement de bit "big-endian" (gros-boutiste) alors que d'autres utilisent le "little-endian" (petit-boutiste).

En général, les en-têtes de DPDU normalisées suivent les conventions de l'IEEE 802.15.4:2011, comme suit:

– La présente norme, excepté les en-têtes DL et MAC, suit les conventions MSB.
– Les en-têtes de DL et MAC normalisés suivent les conventions LSB, avec certaines exceptions clairement indiquées dans l'en-tête de la DPDU.
– Au sein de la DPDU, les sous-en-têtes de sécurité suivent les conventions MSB, suivant le précédent de l'IEEE 802.15.4.
– Tous les champs au sein de l'en-tête de la DPDU sont documentés en montrant le bit 7 à gauche et le bit 0 à droite, suivant la convention de la présente norme.
– Les attributs de DLMO, accessibles au gestionnaire de système par l'intermédiaire du DMAP, sont des informations d'AL et, à ce titre, utilisent généralement la convention MSB. Certaines exceptions sont faites pour des champs qui interagissent directement avec les en-têtes DPDU qui utilisent la convention LSB.

L'ordonnancement des octets LSB est explicitement noté dans diverses parties de la spécification de la DL. Par convention, l'octet 0 est l'octet de poids faible. LSB indique que l'octet de poids faible (octet 0) est émis le premier et que l'octet de poids fort (octet n) est émis le dernier. Sans spécification, l'ordonnancement inverse est utilisé pour l'émission.

9.3.2.2 Nombres entiers non signés de DL extensibles

La spécification de DL utilise une construction appelée ExtDLUint pour l'émission compressée de nombres entiers non signés. Il ne s'agit pas d'un type utilisé dans d'autres normes, et à ce titre, l'ExtDLUint apparaît seulement dans des en-têtes de DPDU et au sein de chaînes d'octets définies par la DL. Il est utilisé pour indiquer le codage compressé d'un nombre entier non signé de 15 bits; il n'est pas utilisé pour acheminer d'autres données. Sachant que ce

type n'est pas utilisé en dehors de la DL, il n'est pas spécifié comme étant un type de données normalisé pris en charge par l'AL.

Un ExtDLUint doit être comme un seul octet lorsque sa valeur se situe dans la plage 0..127, et comme deux octets lorsque sa valeur se situe dans la plage de 128..32 767, avec le codage tel que montré dans le Tableau 108 et dans le Tableau 109. Le bit 0 dans le premier octet indique si un ou deux octets sont émis. L'ordonnancement des octets est toujours tel que montré ici, avec la taille indiquée dans le bit 0 du premier octet émis.

Tableau 108 – ExtDLUint, variante à un seul octet

Octets	Bits							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	2^6	2^5	2^4	2^3	2^2	2^1	2^0	Sélection = 0

Tableau 109 – ExtDLUint, variante à deux octets

Octets	Bits							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	2^6	2^5	2^4	2^3	2^2	2^1	2^0	Sélection = 1
2	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7

9.3.3 En-têtes de la commande d'accès aux supports

9.3.3.1 Généralités

La présente norme utilise un format d'en-tête MAC dont les structures de données sont conformes à la structure détaillée et au codage de champ de l'IEEE 802.15.4:2011, 5.2.1 et 5.2.2.2, suivi d'extensions qui sont particulières à la présente norme. Seules des DPDU Data de l'IEEE 802.15.4:2011, 5.2.2.2 sont utilisées par la présente norme.

La présente norme n'utilise pas la sécurité de l'IEEE 802.15.4. A la place, une sécurité similaire est prise en charge dans le DMXHR. La sécurité de DPDU dans la présente norme est similaire à la sécurité selon l'IEEE 802.15.4; la principale différence est que la présente norme incorpore le sens partagé du temps de la DLE et autres informations spécifiques à un contexte d'utilisation dans les nonces cryptographiques, ce qui conduit à des DPDU plus compactes et à une résistance améliorée contre les attaques de rejet et de changement d'orientation. Pour renforcer la résistance aux attaques, les numéros de séquence DPDU de la présente norme sont dérivés de sources de contenu différentes de celles spécifiées par l'IEEE 802.15.4:2011, 5.2.1.2.

Les DPDU ACK/NAK de l'IEEE 802.15.4:2011, 5.2.3 ne peuvent pas être distinguées de façon fiable d'une DPDU identique temporisée de manière similaire, envoyée par un appareil qui n'est pas le destinataire prévu. Une telle indistinguabilité peut résulter d'une activité concurrente sur la même voie physique par d'autres appareils que le destinataire prévu (par exemple dans d'autres réseaux proches) ou du brouillage délibéré par un attaquant de la DPDU ACK/NAK après réception de la DPDU Data de la transaction. C'est pourquoi la présente norme utilise des DPDU Data authentifiables courtes pour mettre en œuvre des fonctionnalités ACK/NAK similaires mais étendues. Lorsque les adresses MAC source et de destination de telles DPDU ACK/NAK sont les adresses MAC source et de destination, respectivement, de la DPDU Data sollicitante, il n y a pas besoin d'acheminer ces adresses de manière explicite. Ainsi, la présente norme permet à de telles DPDU ACK/NAK de supprimer les deux champs d'adresses MAC, ce qui contredit la contrainte de l'IEEE 802.15.4:2011, 5.2.1.1.8.

NOTE Même si les amendements de l'IEEE 802.15.4 avaient pour but de couvrir des problèmes similaires à ceux qui ont engendré les écarts précédemment décrits, ils le font souvent dans des manières qui sont incompatibles

avec la norme ISA 100.11a:2011 sur laquelle est basée la présente norme internationale et sont donc également incompatibles avec la présente norme, qui s'efforce de maintenir la compatibilité avec les équipements déployés qui utilisent la norme ANSI/ISA en question.

9.3.3.2 En-tête de la commande d'accès aux supports

Le format du sous-ensemble de MHR normalisé est spécifié dans l'IEEE 802.15.4:2011 et l'IEEE 802.15.4:2011, Figure 35, tel qu'utilisé par la présente norme, est résumé dans le Tableau 110.

Tableau 110 – MHR de DPDU Data

Nombre d'octets	bits							
	7	6	5	4	3	2	1	0
2	Frame control (LSB ordering)							
1	Sequence number (Numéro de séquence)							
0 ou 2	PAN ID							
0, 2 ou 8	Adresse de destination							
0, 2 ou 8	Adresse source							
NOTE Le PAN ID, les champs d'adresse de destination et d'adresse source sont chacun émis dans l'ordre LSB.								

La longueur du MHR est habituellement de 9 octets, y compris un PAN ID et deux adresses DL16Address, avec les exceptions suivantes:

- Les DPDU Data de sollicitation ont un PAN ID nul (de longueur égale à zéro) et deux adresses MAC nulles, de telle sorte que le MHR est de 3 octets.
- Les DPDU Data d'annonce ont une adresse MAC de destination nulle, de telle sorte que le MHR est de 7 octets.
- Les autres DPDU Data à destination ou en provenance d'une DLE non rattachée ont une adresse DL16Address et une adresse EUI64Address et, de ce fait, la longueur d'un MHR adressé à destination ou en provenance d'une DLE non rattachée est de 15 octets.

NOTE 1 La capacité de charge utile de la DPDU par défaut, dlmo.MaxDsduSize, est basée sur une taille de MHR de 15 octets et peut être utilisée pour prendre des décisions de fragmentation pour des DLE non rattachées.

- Les DPDU ACK/NAK, qui sont utilisées pour les acquittements immédiats (SCS) ont un PAN ID nul, une adresse MAC de destination nulle, et une adresse MAC source qui est soit
  - nulle (de longueur égale à zéro), de telle sorte que le MHR est de 3 octets; ou
  - une adresse DL16Address, de telle sorte que le MHR est de 5 octets; ou
  - une adresse EUI64Address, de telle sorte que le MHR est de 11 octets.

NOTE 2 Le PAN ID est supprimé car l'authentification de sécurité des DPDU ACK/NAK sert à rejeter toute DPDU ACK/NAK prévue pour un appareil différent, que ce soit sur le même PAN ou un PAN différent.

Conformément au Tableau 110, les champs comprennent:

a) Contrôle de trame. Pour ce champ, les sous-champs sont tels que spécifiés dans l'IEEE 802.15.4:2011, 5.2.1.1:
   - Le FrameType (type de trame) doit être Data.
   - SecurityEnabled doit être "false", car l'IEEE 802.15.4 n'est pas utilisée.

     NOTE 3 La sécurité étendue de la présente norme est prise en charge par DMXHR.
   - FramePending doit être "false".

– Ack.Request doit être "false".

NOTE 4 Le type de DPDU d'acquittement immédiat de l'IEEE 802.15.4 facile à reproduire n'est pas utilisé par la présente norme.

– Lorsqu'une adresse de destination D et une adresse de source D sont toutes les deux incluses, la compression du PAN ID doit indiquer que le même PAN ID est utilisé pour les deux adresses. Le cas échéant, la compression du PAN ID doit être "false" (car une telle compression ne s'applique que lorsque deux adresses D sont présentes).

– Les adresses MAC sont habituellement des adresses DL16Address, avec les exceptions telles que décrites ci-dessous. Les adresses EUI64Address sont utilisées par une DLE se rattachant à un sous-réseau D. Les adresses de destination sont omises dans les annonces dédiées et les sollicitations.

– La version de trame doit être 0x01.

b) Numéro de séquence. Utilisé par le DSC, tel que décrit en 7.3.2.4.10.

NOTE 5 L'IEEE 802.15.4 exige que chaque DLE augmente son numéro de séquence après chaque utilisation afin que le numéro de séquence soit unique pour toutes les DPDU Data et les DPDU ACK/NAK générées par une DLE. Toutefois, la présente norme utilise le champ "sequence number" (numéro de séquence) dans un but quelque peu différent et fournit donc une méthode alternative (autre que la séquentialité cyclique) pour s'assurer que les nonces cryptographiques générés par chaque appareil réel sont uniques au sein de la durée de vie opérationnelle de la clé cryptographique avec laquelle ils sont employés.

NOTE 6 Dans la présente norme, à la fois les DLE et les TLE génèrent des nonces. Les dispositions dont il est fait référence en NOTE 5 s'assurent que les deux ensembles de nonces générés sont disjoints.

c) Le PAN ID doit concorder avec le dlmo.SubnetID et doit être absent des DPDU Data de sollicitation. Si la DPDU achemine à la fois une adresse MAC source et de destination, le PAN ID doit être le PAN ID de la destination (avec le PAN ID source inféré comme étant identique). S'il n'y a pas d'adresse de destination, telle que dans une DPDU Data d'annonce, le PAN ID doit être un PAN ID de source. Dans une DPDU Data de sollicitation, où il n'y a ni source ni adresse de destination, ce champ doit être nul (élidé). Ce champ doit être nul (élidé) dans toutes les DPDU ACK/NAK. Voir 9.1.10.2.

d) L'adresse de destination est normalement un pseudonyme d'adresse DL16Address pour une adresse IPv6Address. Une adresse EUI64Address doit être utilisée pour adresser des DLE qui n'ont pas encore reçu une adresse DL16Address. L'adresse de destination doit être absente dans les DPDU Data d'annonce dédiées, les DPDU Data de sollicitation et les DPDU ACK/NAK.

e) L'adresse source est normalement un pseudonyme d'adresse DL16Address pour une adresse IPv6Address. Une adresse EUI64Address doit être utilisée pour identifier des DLE qui n'ont pas encore reçu une adresse DL16Address. L'adresse de source D doit être absente des DPDU Data de sollicitation, des DPDU ACK/NAK où l'adresse D serait identique à l'adresse D de destination de la DPDU Data reçue qui a initié la transaction.

9.3.3.3 Sous-en-tête DPDU Data

La structure du DHDR pour une DPDU Data est montrée dans le Tableau 111.

Tableau 111 – DHDR de DPDU Data

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	DPDU ACK/NAK attendues	Demande de la qualité du signal dans la DPDU ACK	Demande de l'adresse EUI64Adress dans la DPDU ACK	Inclut DAUX	DMXHR inclut un décalage de saut de voie lent	Destinataire d'horloge	Version DL Toujours 01	

Cette DHDR est toujours de 1 octet.

Conformément au Tableau 111:

- Le bit 7 indique si les DPDU ACK/NAK sont attendues des destinataires adressés de façon explicite ou implicite.
- Le bit 6 indique s'il convient, oui ou non, que la DLE réceptrice rapporte les informations relatives à la qualité de signal dans la DPDU ACK/NAK.
- Le bit 5 indique s'il convient, oui ou non, pour la DLE destinataire d'inclure son adresse EUI64Address dans la DPDU ACK/NAK. Cette valeur de réglage doit être utilisée par l'émetteur chaque fois qu'un acquittement est demandé (bit 7 de valeur 1), et qu'aucune adresse EUI64Address pour le voisin n'existe dans dlmo.Neighbor. Voir 9.1.10.1.

NOTE Le bit 7 est sensé uniquement pour la DPDU Data initiale d'une transaction. Les bits 6 et 5 ne sont sensés que lorsque le bit 7 est sensé et à la valeur true.

- Le bit 4 indique la présence ou l'absence d'un sous-en-tête DAUX dans la DPDU Data.
- Le bit 3 indique si le décalage de saut de voie lent doit être inclus dans le DMXHR. Cette valeur doit être incluse dans les DPDU Data en monodiffusion où le saut de voie lent est utilisé. Voir 9.1.9.2.4.
- Le bit 2 indique si la DLE émettrice est un récepteur d'horloge de DL. Il s'agit d'une demande implicite au récepteur d'inclure une correction d'horloge dans l'acquittement.
- Les bits 0..1 indiquent le numéro de version de DL. Une valeur de 0x01 doit être utilisée, la valeur 0x10 étant réservée pour usage futur. Une valeur de 0x11 est utilisée dans le même emplacement dans la DPDU ACK/NAK et aide à distinguer une DPDU Data d'une DPDU ACK/NAK (voir 9.3.4).

9.3.3.4 Sous-en-tête d'extension MAC de la DPDU

Un DMXHR suivant le DHDR est résumé dans le Tableau 112.

Tableau 112 – DMXHR de DPDU Data

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	Security control							
1	CryptoKeyIdentifier							
0..2	Slow-channel-hopping-offset (ExtDLUint)							

NOTE Pour les futures PHY avec plus de 16 voies, il est vraisemblable que le numéro de voie sera ajouté sous forme de champ virtuel.. Cela sera l'objet d'une normalisation future, mais a été pris en considération dans la conception de DMXHR.

La taille du DMXHR est de 2..4 octets. Une taille de DMXHR de 3 octets, correspondant à un taux de saut de voie lent d'environ 1,25 s ou moins, a été utilisée pour calculer la valeur par défaut de dlmo.MaxDsduSize.

Conformément au Tableau 112, les attributs comprennent:

- Security control et CryptoKeyIdentifier. Les champs de sécurité sont laissés non spécifiés par la DL et sont établis par le DSC. Voir 9.1.12 pour une vue d'ensemble de la relation entre la DL et les DSC. Alors que l'IEEE permet un attribut CryptoKeyIdentifier d'une longueur pouvant atteindre 9 octets, sa taille est toujours égale à 1 octet dans la présente norme.
- Le décalage de saut de voie lent spécifie le décalage de l'intervalle de temps au cours de la période de saut de voie lent, s'il est nécessaire d'identifier sans ambiguïté un intervalle de temps (car l'initiateur de transaction et le répondeur ont des perceptions locales différentes du même intervalle de temps). La présence ou l'absence de ce champ est indiquée dans le DHDR. Le décalage de saut de voie lent est décrit en 9.1.9.4.9.

9.3.3.5 Sous-en-tête auxiliaire de DPDU

Le sous-en-tête DAUX est utilisé pour:

- la découverte de voisins de DL;
- l'activation temporaire de liaisons;
- les rapports relatifs à la qualité de signal reçu dans les acquittements.

Le sous-en-tête DAUX, présent seulement lorsque le bit 4 de l'octet du DHDR est mis, est décrit en 9.3.5.

Une longueur de DAUX de 0 octet a été utilisée pour calculer la valeur par défaut de dlmo.MaxDsduSize.

NOTE dlmo.MaxDsduSize est utilisé pour prendre des décisions de fragmentation. Le sous-en-tête DAUX est utilisable pour activer des liaisons dans un scénario de fragmentation. Cependant, l'activation de liaison n'est pas possible au cours du processus de rattachement et, à ce titre, l'activation de liaison n'est jamais combinée avec les adresses EUI64Address. Parce que le calcul de dlmo.MaxDsduSize inclut une adresse EUI64Addresses, il permet la prise en charge d'un sous-en-tête d'activation de liaison DAUX lorsqu'une adresse EUI64Addresses n'est pas présente.

9.3.3.6 Sous-en-tête de routage de DPDU

Il existe deux variantes du sous-en-tête DROUT. Une variante compressée, de taille égale à deux octets, est utilisée lorsqu'un seul graphe est utilisé pour l'adressage. La variante compressée est également utilisée lorsque le routage à un seul saut est utilisé, le chemin étant implicite dans l'adressage de niveau MAC donné dans le MHR selon l'IEEE 802.15.4. Lorsqu'une série d'adresses est nécessaire, une variante non compressée du sous-en-tête DROUT doit être utilisée.

Le sous-en-tête DROUT doit être élidé dans une DPDU Data qui n'a pas de charge utile de couche supérieure, comme l'indique une DSDU de taille zéro.

La variante compressée du sous-en-tête DROUT doit être utilisée dans le cas commun où un seul graphe, avec un indice de 255 ou moins, est utilisé pour le routage. Elle est montrée dans le Tableau 113.

Tableau 113 – Structure de DROUT, variante compressée

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	Compress=1	Priority				DlForwardLimit		
0..1	DlForwardLimitExt							
1	GraphID							

Une taille de DROUT de 2 octets a été utilisée pour calculer la valeur par défaut de dlmo.MaxDsduSize. MaxDsduSize a normalement besoin d'être réduit lorsque le routage source est utilisé.

Conformément au Tableau 113, la variante compressée du DROUT comprend:

- Compress. Si cette valeur est mise à 1, la variante compressée du format de DROUT doit être utilisée.
- Priority. Elle doit être mise à la priorité de 4 bis de la DPDU Data.
- DlForwardLimit et DlForwardLimitExt (limite de transmission) limitent le nombre de fois qu'une DPDU Data peut être transmise au sein d'un sous-réseau D. Si la limite de transmission est inférieure à 7, la valeur doit être transmise dans DlForwardLimit et DlForwardLimitExt doit être élidé. Si la limite des transmissions est supérieure ou

égale à 7, DlForwardLimit doit être émis comme étant 7, et la limite des transmissions doit être émise dans DlForwardLimitExt.

La limite des transmissions est initialisée par la DL lorsque le chemin est sélectionné, en fonction de la valeur de dlmo.Route[ ].ForwardLimit. Lorsqu'une DPDU Data en monodiffusion est reçue avec succès par la DL et a besoin d'être transmise, la DPDU Data doit être rejetée si sa limite de transmissions est zéro. Si sa limite de transmissions est positive, la limite de transmissions doit être décrémentée (éventuellement à zéro) et la DPDU Data doit être placée dans la file d'attente de messages.

- GraphID (8 bits). Les GraphID conformes à la présente norme sont des nombres entiers non signés de 12 bits. Dans le cas commun où le chemin est un seul ID de graphe dans la plage 1..255, la variante compressée du sous-en-tête DROUT doit être utilisée. En outre, la variante compressée est utilisée dans le routage de source à un seul saut, dans lequel GraphID=0 doit indiquer que la destination est située à une distance d'un saut. Sachant que l'adresse de destination à un seul saut peut être trouvée dans le MHR, elle peut ne pas être répétée dans DROUT. La valeur GraphID=0 doit être utilisée au cours du processus de rattachement pour un adressage à destination et en provenance d'un proxy voisin, et elle représente la seule manière dans la présente norme d'indiquer une adresse de destination EUI64Address dans DROUT.

NOTE il est possible pour un gestionnaire de système de configurer un chemin D circulaire, la DPDU Data étant transmise jusqu'à ce que ForwardLimit décrémente à zéro. Le champ LH dans l'en-tête de DL, décrit en 9.3.3.7, n'est pas destiné à empêcher les chemins circulaires au sein d'un sous-réseau D.

La variante non compressée du sous-en-tête DROUT est montrée dans le Tableau 114.

Tableau 114 – Structure de DROUT, variante non compressée

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	Compress=0	Priority				DlForwardLimit		
0..1	DlForwardLimitExt							
1	*N* (nombre d'entrées dans la table de routage)							
2*N	Séries de *N* GraphIDs/addresses (Unsigned16, LSB)							
NOTE Chaque GraphID/address est transmise dans l'ordre LSB.								

Conformément au Tableau 114, la variante non compressée du sous-en-tête DROUT comprend:

- Compression: Si cette valeur est mise à 0, la variante non compressée du format de DROUT doit être utilisée.
- Priority, DlForwardLimit, et DlForwardLimitExt sont tels que décrits ci-dessus avec le Tableau 113.
- *N*: Ce champ doit être mis au nombre d'entrées dans Route. Les entrées peuvent être une combinaison de GraphID et d'adresses D DL16Address. La valeur de *N* ne doit pas dépasser 15.
- Route: Ce champ doit être mis à une série de GraphID et/ou d'adresses DL16Address, indiquant le chemin, dans l'ordre, le long duquel la DPDU Data se déplacera. L'IETF RFC 4944 limite las plages d'adresses en monodiffusion en plages $1..2^{15}-1$. Les GraphIDs 12 bits de ce champ doivent être représentés de manière disjointe du rang d'adresse 0x 1010 gggg gggg gggg, qui est le rang $10\times2^{12}..11\times2^{12}-1$.

Lorsque le routage de source est utilisé, le sous-en-tête DROUT doit être raccourci par la DL des routeurs intermédiaires à mesure que la DPDU Data progresse le long du chemin, comme décrit en 9.1.6.

La première entrée dans le sous-en-tête DROUT est utilisée pour déterminer le prochain saut. Par exemple, le chemin peut être spécifié à la source comme étant <000 123, 000 456, 000 789>. L'adresse de premier saut, <000 123>, est utilisée pour envoyer la DPDU Data vers un voisin immédiat. Le sous-en-tête DROUT, tel que reçu par la DLE <000 123>, contient la route source <000 123, 000 456, 000 789>. Lorsqu'il est reçu, ce chemin est raccourci à <000 456, 000 789> (voir 9.1.6.3), indiquant que l'adresse <000 456> est le prochain saut.

Lorsqu'un graphe est spécifié comme étant la première entrée dans un chemin de source, la DPDU Data doit suivre le graphe jusqu'à ce qu'il soit terminé comme décrit en 9.1.6.

9.3.3.7 Sous-en-tête d'adressage

Le sous-en-tête d'adressage (DADDR) inclut les adresses de source et de destination de NL, avec trois champs de NL qui sont visibles à la DL.

Le sous-en-tête DADDR doit être élidé dans une DPDU Data qui n'a pas de charge utile d'ordre supérieur, à savoir une DSDU de taille zéro.

La structure du sous-en-tête DADDR est montrée dans le Tableau 115.

Tableau 115 – Structure de DADDR

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	DE	LH	ECN		Reserved=0			
1..2	SrcAddr (ExtDLUint)							
1..2	DestAddr (ExtDLUint)							

Une taille de DADDR de 4 octets a été utilisée pour calculer la valeur par défaut de dlmo.MaxDsduSize, reflétant une hypothèse selon laquelle une adresse ou les autres adresses est (sont) encodée(s) en un octet.

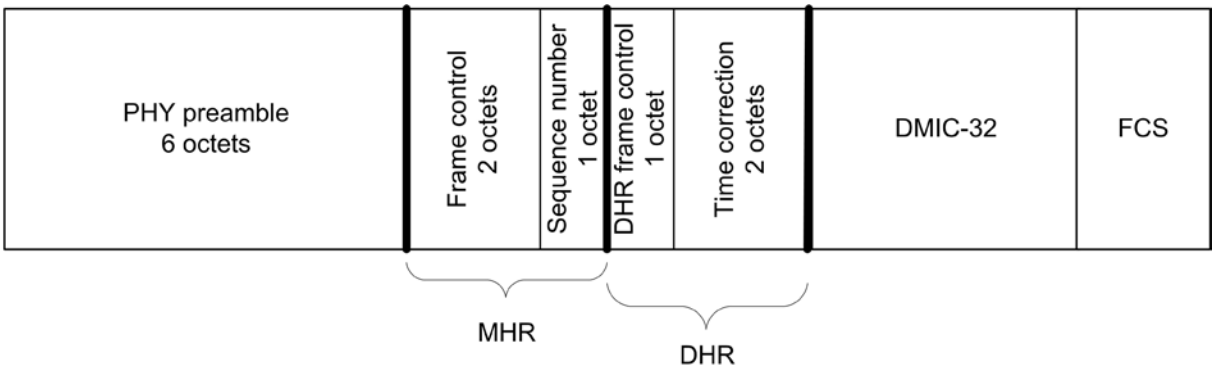
Les champs comprennent:

- Le champ Discard eligible (DE) doit être mis en fonction de la valeur fournie par la NL par l'intermédiaire de DD-DATA.request. DE=1 indique que la DSDU est éligible à être rejetée de la file d'attente de messages en faveur d'une DPDU Data entrante avec DE=0, et de priorité égale ou plus haute.
- Le champ LastHop (LH) doit être mis en fonction de la valeur fournie par la NL par l'intermédiaire de DD-DATA.request. Ce bit est transporté par la DL pour éviter les chemins circulaires à la NL, et n'affecte pas le comportement de la DL.
- Le champ ExplicitCongestionNotification (ECN) doit être mis en fonction de la valeur fournie par la NL par l'intermédiaire de DD-DATA.request. Un routeur subissant l'encombrement peut positionner l'ECN comme décrit dans l'IETF RFC 3168. Voir 9.1.9.4.5 pour un débat relatif à l'ECN.
- SrcAddr est mis en fonction de la valeur fournie par la NL par l'intermédiaire de DD-DATA.request. Si l'adresse D source de est dupliquée dans le champ adresse D de source de MHR, SrcAddr doit être encodé 0x00. Cela couvre le cas, au cours du processus de rattachement, dans lequel l'adresse source est une adresse EUI64Address.
- DestAddr est mis en fonction de la valeur fournie par la NL par l'intermédiaire de DD-DATA.request. Si l'adresse D de destination est dupliquée dans le champ d'adresse de destination de MHR, DestAddr doit être encodé 0x00. Cela couvre le cas, au cours du processus de rattachement, dans lequel l'adresse D de destination est une adresse EUI64Address.

En codant une adresse DL dupliquée comme étant zéro, un octet est compressé sur le premier saut et sur le dernier saut lorsque l'adresse est inférieure à 0x0080, économisant de l'énergie. Ainsi, une adresse de DADDR codée comme étant zéro référence l'adresse DL16Address ou l'adresse EUI64Address correspondante dans le MHR.

9.3.4 Les DPDU d'acquittement MAC

La Figure 89 montre la structure des DPDU ACK/NAK.



NOTE This figure includes only the most commonly used fields.

Légende

Anglais	Français
PHY preamble 6 octets	Préambule PHY 6 octets
Frame control 2 octets	Contrôle de trames 2 octet
Sequence number 1 octet	Numéro de séquence 1 octet
DHR frame control 1 octet	Contrôle de trame DHR 1 octet
Time correction 2 octets	Correction de temps 2 octets
NOTE This figure includes only the most commonly used fields	NOTE Cette figure n'inclut que les champs les plus communément utilisés

Figure 89 – Disposition type des DPDU ACK/NAK

L'adresse source d'une DPDU ACK/NAK est l'adresse de la DLE qui émet la DPDU. L'adresse de destination est l'adresse du destinataire prévu de la DPDU.

Chaque DPDU ACK/NAK doit être authentifiée avec un DMIC, mais pas chiffrée. Certains champs sont virtuels, utilisés pour créer le DMIC, mais pas réellement émis.

Bien que la DPDU ACK/NAK soit une trame de donnée de l'IEEE 802.15.4, elle peut être distinguée des autres trames de données de type IEEE 802.15.4 en se basant sur:

– la temporisation de la DPDU ACK/NAK, suivant l'émission de la DPDU, telle que spécifiée en 9.4.3.3; et
– Les bits 1..0 de son champ de contrôle de trame de DHR, tel que spécifié dans le Tableau 118; et
– un champ virtuel dans la DPDU ACK/NAK, qui fait écho au DMIC original de la DPDU Data, tel que spécifié dans le Tableau 117.

Comme décrit en 7.3.2.2, le DMIC dans une DPDU ACK/NAK utilise la même politique de sécurité que la DPDU Data d'origine de la transaction D dont cet ACK/NAK est la réponse , avec l'exception que la taille des DPDU ACK/NAK de DMIC doit toujours être de 32 bits, quelle que soit la politique de sécurité des DPDU Data.

Le format d'un MHR selon l'IEEE 802.15.4 est résumé dans le Tableau 116.

Tableau 116 – MHR de DPDU ACK/NAK

Nombre d'octets	Bits							
	7	6	5	6	3	2	1	0
2	Frame control (LSB ordering)							
1	Sequence number (Numéro de séquence)							
0	Adresse de destination (nulle)							
0 ou 2	PAN ID							
0, 2 ou 8	Adresse source							
NOTE Le PAN ID et les champs d'adresse source sont chacun émis dans l'ordre LSB.								

La description détaillée de ces champs est spécifiée dans l'IEEE 802.15.4. Conformément au Tableau 116, les attributs comprennent:

- Les attributs de commande de trame (Frame control) pour les DPDU ACK/NAK, comme suit:
  - Le FrameType (type de trame) doit être Data.
  - SecurityEnabled doit être désactivée, car elle est traitée dans le DHR.
  - FramePending doit être "false".
  - AckRequest doit être "false".

  NOTE 1 Le bit ci-dessus demande la génération de la forme non sécurisable de l'acquittement immédiat offert par l'IEEE 802.15.4:, qui n'est pas utilisée par la présente norme.

  - Le mode d'adressage de source doit être 0x00 (à savoir: implicite), excepté pour les cas décrits ci-dessous dans lesquels le PAN ID et l'adresse source sont inclus dans le MHR.
  - Le mode d'adressage de destination doit être 0x00 (à savoir: implicite).
  - FrameVersion doit être 0x01.
- Numéro de séquence, utilisé par le DSC, tel que décrit en 7.3.2.4.10. Sachant que la présente norme n'utilise pas le type de trame ACK non sécurisable spécifié par l'IEEE 802.15.4, la DPDU ACK/NAK ne transporte pas le numéro de séquence issu de la DPDU Data précédente. A la place, le numéro de séquence doit être lui-même similaire à celui d'une DPDU Data (voir 9.3.3.2) et doit être utilisé dans la construction du nonce D pour le DMIC de la DPDU ACK/NAK.
- PAN ID, présent uniquement lorsque l'adresse source est présente (non nulle).
- Adresse source. Normalement, une adresse D source n'est pas incluse dans une DPDU ACK/NAK parce qu'elle concorde avec l'adresse D de destination de la dernière DPDU Data reçue. Cependant, il existe deux exceptions où elle est incluse:
  - Un acquitteur immédiat d'une DPDU Data reçue doit inclure son adresse EUI64Address en tant qu'adresse source du MHR de la DPDU ACK/NAK de réponse, lorsque cela est demandé dans le DHDR de la DPDU Data reçue.
  - Un acquitteur immédiat d'une DPDU Data reçue dont l'adresse D est différente de l'adresse D de destination de la dernière DPDU Data reçue doit inclure son adresse DL16Address en tant qu'adresse source de la DPDU ACK/NAK de réponse.

  NOTE 2 Cette seconde exception se produit dans les acquittements duodiffusion/N-diffusion secondaires.

Un prototype de DHR suivant un MHR est résumé dans le Tableau 117.

Tableau 117 – DHR de DPDU ACK/NAK

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	ACK/NAK DPDU DHDR							
4 (virtuels)	DMIC en écho de la DPDU Data reçue							
0, 2	Correction du temps (Unsigned16, LSB) lorsque demandée							
0..2	Décalage d'intervalle de temps (ExtDLUint) lorsque nécessaire							
0..3	Sous-en-tête DAUX habituellement absent							

Conformément au Tableau 117, les attributs comprennent:

- Le DHDR de la DPDU ACK/NAK est décrit dans le Tableau 118.
- DMIC en écho de la DPDU Data reçue. Pour un débat relatif au traitement de ce champ virtuel, voir 7.3.2. Pour connecter sans ambiguïté la DPDU ACK/NAK à la DPDU Data à laquelle elle est une réponse, le DMIC de la DPDU Data est inclus dans le DHR de la DPDU ACK/NAK comme un champ virtuel, avec l'ordonnancement d'octets concordant au DMIC de la DPDU Data. Ce champ virtuel est utilisé pour calculer le DMIC de la DPDU ACK/NAK, mais il n'est pas émis. Si le DMIC reçu a une longueur de plus de 4 octets, seuls les 4 octets initiaux (les plus à gauche) du DMIC sont repris en écho comme un champ virtuel.
- Correction de temps (LSB). Utilisée par des sources d'horloge de DL pour corriger le temps du destinataire d'horloge de DL, si elle est demandée dans le DHDR de la DPDU reçue. Cette valeur non signée de 2 octets, lorsqu'elle est incluse dans la DPDU ACK/NAK, fait écho au temps auquel la DPDU Data a été reçue. La valeur, en 2^{-20}s (approximativement 0,954 μs), rapporte un décalage par rapport au temps de début programmé de l'intervalle de temps courant dans la base de temps de l'acquitteur. La valeur rapportée est basée sur le temps de début de la DPDU Data. Voir 9.1.9.3.2.
- Le décalage d'intervalle de temps de l'acquitteur est fourni si nécessaire au sein d'une période de saut de voie lent. Cette valeur, lorsqu'elle est incluse dans la DPDU ACK/NAK, indique l'intervalle de temps courant dans la base du temps de l'acquitteur. Elle doit être incluse seulement lorsque la DPDU Data est reçue dans un intervalle de temps de saut de voie lent différent de celui utilisé dans l'acquittement. Le premier intervalle de temps dans une période de saut de voie lent a un décalage de zéro. Lorsque le décalage corrigé d'intervalle de temps est non nul, la correction de temps (champ précédent), lorsqu'elle est incluse, doit être la valeur corrigée du temps programmé d'intervalle de temps. La sécurité exige que le temps d'une DLE augmente d'un intervalle de temps à l'autre. Par conséquent, si l'intervalle de temps est corrigé à un intervalle de temps plus précoce par un destinataire d'horloge, il doit y avoir une interruption de service, égale à la grandeur de la correction d'intervalle de temps plus au moins un intervalle de temps. Voir 9.1.9.4.9.
- Sous-en-tête auxiliaire (DAUX). Le DAUX peut être inclus dans une DPDU ACK/NAK, dans le but limité de reprendre en écho la qualité de signal reçu (voir 9.3.5.5).

Dans une DPDU ACK/NAK, l'octet DHDR communique le type d'ACK/NAK et d'autres informations relatives à la sous-structure de la DPDU (voir Tableau 118).

Tableau 118 – DHDR d'une DPDU ACK/NAK

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	Correction d'horloge incluse	Décalage de saut de voie lent inclus	Type d'ACK/NAK: 0: ACK 1: ACKwithECN 2: NAK0 3: NAK1		Sous-en-tête DAUX inclus	Reserved (=0)	DPDU ACK/NAK (=11)	

Le numéro de version de protocole de DL et la clé de sécurité MAC concordent toujours avec la DPDU Data reçue à laquelle la DPDU ACK/NAK est un acquittement immédiate; ils ne sont donc pas indiqués explicitement dans la DPDU ACK/NAK.

Le contenu en bits est comme suit:

- Le bit 7 indique si la DPDU ACK/NAK inclut des informations relatives à la correction d'horloge.
- Le bit 6 indique si la DPDU ACK/NAK inclut un décalage de saut de voie lent.
- Les bits 5..4 indiquent la classe de la DPDU ACK/NAK:

  0b10: un acquittement négatif NAK0 signalant que la DPDU Data a été reçue, mais qui ne pourrait pas être acquittée en raison d'un encombrement de la file d'attente de messages (9.1.9.4.4);

  0b11: un NAK1, signalant que la DPDU Data a été reçue, mais n'a pas été acceptée en raison d'antécédents récents de problèmes de transmission le long du chemin (9.1.9.4.4);

  0b00: un acquittement positif ACK;

  0b01: un acquittement positif ACK avec une notification d'encombrement explicite (ECN) (9.1.9.4.5).

  Il convient qu'un routeur qui signale une ECN dans le sens direct signale également l'ECN par des DPDU ACK/NAK lorsque la priorité de la DPDU Data est 7 ou moins. Une DLE recevant une ECN par l'intermédiaire d'une DPDU ACK/NAK peut traiter ce signal comme une notification précoce qu'il est susceptible de recevoir une ECN à des couches supérieures;
- Le bit 3 indique si la DPDU ACK/NAK inclut un sous-en-tête DAUX qui peut être inclus dans une DPDU ACK/NAK dans le but limité de rapporter la qualité de signal reçu.
- Le bit 2 est réservé et doit être mis à zéro.
- Les bit 1..0 sont mis à des uns partout (11) pour distinguer les DPDU ACK/NAK des autres DPDU.

9.3.5 Sous-en-tête auxiliaire de DL

9.3.5.1 Généralités

Un sous-en-tête auxiliaire (DAUX) peut être inclus dans n'importe quelle DPDU Data ou DPDU ACK/NAK. Les bits 7..5 du premier octet de DAUX déterminent son type, avec le format de sous-en-tête subséquent différent pour chaque type. Les types définis sont:

- Advertisement: type 0 dans la DPDU Data: fournit des informations nécessaires à de nouvelles DLE pour se synchroniser avec le sous-réseau D et le rejoindre;
- Solicitation: type 1 dans la DPDU Data: sollicite une annonce issue d'une DLE voisine;
- Activate link: type 2 dans la DPDU Data: active une liaison inactive pendant une période de temps;
- Signal quality: type 3 dans la DPDU ACK/NAK: rapporte la qualité du signal reçu.

Toutes les combinaisons de type et de classe DPDU sont réservées pour usage futur.

NOTE Suivant les conventions d'en-tête de DL, les champs DAUX utilisent l'ordre LSB ("little-endian", petit-boutiste) pour l'émission. Il existe certaines structures similaires dans le DLMO qui utilisent l'ordre MSB ("big-endian", gros boutiste). Par exemple, des structures de supertrame sont spécifiées dans les deux endroits, en utilisant l'ordre LSB dans l'en-tête de DL et l'ordre MSB dans des attributs du DLMO.

9.3.5.2 Sous-en-tête auxiliaire de type annonce (advertisement)

9.3.5.2.1 Généralités

Les champs au sein d'un DAUX d'annonce peuvent être groupés logiquement comme:

- sélection d'annonce;
- synchronisation du temps
- informations de supertrame;
- informations de rattachement; et
- contrôle d'intégrité

Le Tableau 119 résume la structure du DAUX d'annonce.

Tableau 119 – Structure du DAUX d'annonce

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	Sélection d'annonce; voir Tableau 120							
6	Synchronisation du temps; voir Tableau 122							
6..10	Informations de supertrame; voir Tableau 124							
4..10	Informations de rattachement; voir Tableau 127							
2	Contrôle d'intégrité; voir 9.3.5.2.4.4							
NOTE Telle que décrite en 9.3.5.2.4.2, la taille du champ informations de rattachement est limitée à 10 octets.								

L'ID sous-réseau D du routeur d'annonce et l'adresse DL16Address sont acheminés par la sous-couche MAC, et peuvent ne pas être émis de façon redondante au sein du DAUX.

Un DAUX d'annonce peut être inclus au sein d'une DPDU Data, mais ne doit être pas inclus dans une DPDU ACK/NAK.

Une annonce inclut des informations qui permettent à la DLE réceptrice de créer des supertrames et des liaisons devant être utilisées au cours du processus de rattachement. Ces informations doivent être conservées par la DLE à la fin du processus de rattachement et, avec des valeurs par défaut de DL, constituer une base de données de départ pour les informations de planification de liaisons de la DLE. Les mêmes liaisons utilisées pour le rattachement sont temporairement utilisées pour des communications générales jusqu'à ce que le gestionnaire de système fournisse une configuration de remplacement.

Les attributs mis par la DLE en fonction des informations figurant dans l'annonce reçue incluent:

- dlmo.SubnetID est positionné en fonction du SubnetID dans l'annonce.
- TAI time (c'est-à-dire temps TAI) est synchronisé par l'annonce.
- dlmo.Superframe number1 est créé avec des champs copiés à partir de l'annonce.
- dlmo.Link number1 est créé comme une liaison d'émission avec des champs copiés à partir de l'annonce.
- dlmo.Link number2 est créé comme une liaison de réception avec des champs copiés à partir de l'annonce.

- dlmo.Link number3 peut être créé comme des liaisons de réception à balayage passif avec des champs copiés à partir de l'annonce, si fournis.
- dlmo.Neighbor est initialisé par la DLE avec une entrée correspondant au routeur d'annonce.
- dlmo.Graph number1 est automatiquement créé par la DLE pour fournir un accès au routeur d'annonce.
- dlmo.Route number1 est automatiquement créé par la DLE comme chemin par défaut utilisant le graphe numéro 1.

9.3.5.2.2 Sélections d'annonces

Le Tableau 120 spécifie le champ sélections d'annonces dans le DAUX d'annonce.

Tableau 120 – Eléments de sélections d'annonce

Nom de l'élément	Codage d'élément
DauxType	Type: Unsigned3 0=DAUX d'annonce
ChMapOv	Type: Unsigned1 0=default
DauxOptSlowHop	Type: Unsigned1 0=default
Reserved (alignement d'octets)	Type: Unsigned3=0

Le Tableau 121 montre la structure du champ sélections d'annonces.

Tableau 121 – Sélections d'annonces

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	DauxType=0			DauxChMapOv	DauxOptSlowHop	Reserved=0		

Le champ sélections d'annonces fait 1 octet. Conformément au Tableau 120, les attributs comprennent:

- DauxType. Toujours mis à 0 pour un DAUX d'annonce. Indique la structure du DAUX dans le Tableau 119.
- DauxChMapOv. La valeur true indique que le champ DauxChMap est inclus dans la DPDU d'annonce. La valeur false sélectionne la carte de voies par défaut 0x7FFF. Ce champ correspond à dlmo.Superframe[ ].ChMapOv.
- DauxOptSlowHop. La valeur true indique que le sous-réseau D offre le saut de voie lent et que le champ DauxChMap est inclus dans la DPDU d'annonce. La valeur false indique la valeur par défaut pour le saut de voie en fonction du temps.

NOTE Le saut de voie lent peut être utilisé durant le processus de rattachement, mais aussi après.

- Les bits 2..0 sont réservés et doivent être mis à 0.

9.3.5.2.3 Synchronisation du temps d'annonce

Le Tableau 122 spécifie le champ synchronisation du temps dans le DAUX d'annonce.

Tableau 122 – Eléments de synchronisation du temps d'annonce

Nom de l'élément	Codage d'élément
DauxTAIsecond (temps TAI courant)	Type: Unsigned32 (LSB) Unités: 1 s
DauxTAIfraction (seconde TAI fractionnaire)	Type: Unsigned16 (LSB) Unités: 2^{-15} s

Le Tableau 123 montre la structure du champ synchronisation du temps d'annonce.

Tableau 123 – Structure de synchronisation du temps d'annonce

Octets	Bits								
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Interprétation**
1	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	DauxTAIsecond; Partie intégrale du temps TAI avec une granularité de 1 s
2	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	
3	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
4	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	
5	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}	DauxTAIfraction; Partie fractionnelle du temps TAI avec une granularité de 2^{-15} s
6	0 réservé	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	
NOTE La représentation ci-dessus est radicalement différente de celle de TAINetworkTime (Table 362), l'ordre d'octets étant inversé dans les quatre premiers octets et l'ordre d'octets étant à la fois inversé et décalé d'un bit dans les deux derniers octets.									

NOTE 1 DauxTAIfraction de 2^{-15} s a été retenu pour correspondre aux verres "de montre" très faible puissance et très précis de 32 kiHz qui sont généralement utilisés pour le matériel d'horloge continu des appareils WISN.

Le champ synchronisation du temps fait 6 octets. Conformément au Tableau 122, les sous-champs comprennent:

– DauxTAIsecond. Temps TAI courant en unités de 1 s.

– DauxTAIfraction. Seconde TAI fractionnaire en unités de 2^{-15} s, avec une plage de 0..32 667. Dans la seconde TAI, cela indique le temps de départ réel de la DPDU d'annonce. (Il convient qu'une mise en œuvre qui cadence réellement en fonction de la temporisation de SFD rende compte d'un temps de départ de DPDU qui est nominalement 1 octet, ou 32 μs, plus tardif que le temps où le délimiteur de début de trame, le SFD, est complètement émis/reçu.)

NOTE 2 Même si le temps TAI est normalement représenté sous la forme d'une fraction binaire à point fixe mise à l'échelle de 6 octets, modulo 2^{32} s, l'ordre d'octets de transmission à deux parties ci-dessus, où chaque partie est transmise séparément dans l'ordre LSB en premier et où la partie fractionnaire comporte un bit inutilisé inséré au point binaire, ne respecte pas l'ordre d'octets naturel de cette fraction à point fixe mise à l'échelle.

Voir 9.1.9 pour plus d'informations relatives au temps TAI et à l'alignement d'intervalles de temps.

L'identité et la temporisation programmée de l'intervalle de temps courant peuvent être dérivées du temps de départ réel de la DPDU Data combiné à la description de supertrame de rattachement. Voir 9.1.9.1.5.

Le temps dans une annonce doit être un reflet exact de l'horloge TAI interne de l'annonceur à ±96 μs (la durée de transmission de 3 octets) près, ce qui correspond à la fenêtre d'émission présumée pour les appareils conformes à la présente norme.

Une DPDU ACK/NAK en réponse à une demande de rattachement inclut une correction d'horloge qui peut être plus précise que l'annonce initiale, et également plus courante.

9.3.5.2.4 Supertrame et liaisons de rattachement d'annonce

9.3.5.2.4.1 Supertrame de rattachement d'annonce

NOTE Le processus de rattachement, y compris les sollicitations, les annonces et l'utilisation des informations transmises dans les annonces, est décrit en 7.4.

Trois liaisons sont spécifiées par l'annonce relativement à la découverte de voisins:

- liaison numéro 1 pour envoyer des demandes de rattachement, adressées au routeur d'annonce voisin;
- liaison numéro 2 pour recevoir les réponses de rattachement consécutives issues du routeur d'annonce; et
- liaison numéro 3 pour rechercher par balayage des voisins supplémentaires après que la DLE a réussi à rejoindre le sous-réseau D.

Toutes ces liaisons se réfèrent à la supertrame number1, qui est également spécifiée dans l'annonce.

Les noms de champ dans l'annonce correspondent aux champs équivalents dans dlmo.Superframe et dlmo.Link. Suivant les conventions d'en-tête de DL, l'ordonnancement d'octet LSB est utilisé sur certains champs qui sont émis en utilisant l'ordonnancement MSB dans la supertrame elle-même. Pour réduire au maximum les exigences relatives au traitement et compresser le sous-en-tête DAUX, un sous-ensemble de caractéristiques de supertrame et de liaison est pris en charge par l'intermédiaire de l'annonce.

Le Tableau 124 spécifie le champ d'informations de supertrame de rattachement.

Tableau 124 – Sous-champs d'informations de supertrame de rattachement

Nom de sous-champ	Codage de sous-champ
DauxTsDur (durée d'intervalle de temps)	Type: Unsigned16 Unités: 2^{-20} s
DauxChIndex (ID de modèle de saut de voie)	Type: ExtDLUint Plage valide: 1..5
DauxChBirth (point de début de référence de saut de voie)	Type: Unsigned8
DauxSfPeriod (nombre de base d'intervalles de temps dans chaque cycle de supertrame)	Type: ExtDLUint
DauxSfBirth (point de début du cycle de supertrame)	Type: ExtDLUint Plage valide: 0..127
DauxChRate (longueur de chaque période de saut de voie lent, en nombre d'intervalles de temps)	Type: Unsigned8 Non émis et prenant la valeur par défaut 1 lorsque DauxOptSlowHop est false
DauxChMap (carte des voies à saut de voie pour la gestion de spectre)	Type: Unsigned16 (LSB) Non émis et prenant la valeur par défaut 0x7FFF lorsque advChMapOv est false

Le Tableau 125 résume la structure du champ "informations de supertrame de rattachement" dans le DAUX d'annonce. Les champs ExtDLUint y sont montrés comme étant un seul octet.

Tableau 125 – Structure d'informations de supertrame de rattachement

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
2	DauxTsDur (LSB)							
1	DauxChIndex							
1	DauxChBirth							
1..2	DauxSfPeriod							
1	DauxSfBirth							
0..1	DauxChRate							
0..2	DauxChMap (LSB)							

Le champ informations de supertrame de rattachement fait 6..10 octets.

Chaque sous-champ d'informations de supertrame de rattachement d'annonce correspond à un champ dans dlmo.Superframe. Voir 9.4.3.5.

Pour créer la supertrame number1 à partir de l'annonce, la DL utilise des valeurs issues de l'annonce pour initialiser les champs de supertrame correspondants. Les champs dans la supertrame number1 qui n'ont pas des champs nommés d'une manière équivalente dans l'annonce prennent par défaut des valeurs fixes. Le Tableau 126 montre le mapping issu des sous-champs de l'annonce à la supertrame number1.

Tableau 126 – Supertrame dérivée de l'annonce

Nom du champ de supertrame	Valeur	Notes
* Index	1	-
TsDur	DauxTsDur	-
ChIndex	DauxChIndex	-
ChBirth	DauxChBirth	-
SFType	0	-
Priority	0	-
ChMapOv	DauxChMapOv	-
IdleUsed	0	-
SfPeriod	DauxSfPeriod	Type de données compressé utilisé dans l'annonce. Limite les supertrames utilisées pour le rattachement à une période de 300 s approximativement.
SfBirth	DauxSfBirth	Type de données compressé utilisé dans l'annonce, compatible avec SfPeriod
ChRate	DauxChRate	Type de données compressé utilisé dans l'annonce. Limite les supertrames utilisées pour le rattachement à une fréquence de saut de voie lent approximativement égale à un saut toute les 2,5 s.
ChMap	DauxChMap	Convertit les octets LSB en MSB
IdleTimer	null	-
rndSlots	null	-

9.3.5.2.4.2 Liaisons de rattachement d'annonce

NOTE 1 Le processus de rattachement, y compris les sollicitations, les annonces et l'utilisation des informations transmises dans les annonces, est décrit en 7.4.

Deux jeux de liaisons relatives au rattachement sont fournis dans chaque annonce:

- un jeu sortant de liaisons pour émettre des demandes de rattachement, utilisé pour initialiser dlmo.Link number1; et
- un jeu entrant de liaisons pour recevoir des réponses de rattachement, utilisé pour initialiser dlmo.Link number2;

en outre, le routeur d'annonce peut fournir un jeu de liaisons qu'une DLE peut utiliser pour rechercher par balayage des annonces lorsque le rattachement est achevé, utilisé pour initialiser dlmo.Link number3 lorsqu'il est fourni.

Chaque appareil qui tente de rejoindre un sous-réseau, à la réception d'une annonce d'un sous-réseau D qu'il choisit de rejoindre, doit configurer ses liaisons entrantes et sortantes en fonction des informations figurant dans l'annonce reçue. Il doit ensuite transmettre sa demande de rattachement au routeur d'annonce en utilisant ces liaisons sortantes.

Les liaisons utilisées pour le rattachement sont contraintes par rapport à un jeu élémentaire de caractéristiques. Des modèles d'intervalle de temps par défaut sont utilisés; voir Tableau 165, Tableau 166, et Tableau 167.

Trois types de liaisons sont identifiés:

– les liaisons JoinTx sont utilisées pour émettre des demandes de rattachement vers le routeur d'annonce;
– les liaisons JoinRx sont surveillées pendant l'attente d'une réponse de rattachement;
– les liaisons AdvRx, lorsqu'elles sont fournies, sont activées lorsque le rattachement est achevé pour rechercher par balayage passif des annonces issues de routeurs de remplacement.

Le Tableau 127 spécifie le champ informations de rattachement dans le DAUX d'annonce.

Tableau 127 – Eléments d'informations de rattachement

Nom de l'élément	Codage d'élément
DauxJoinBackoff (étendue maximale du repli et de répétition de tentative lors du rattachement)	Type: Unsigned4
DauxJoinTimeout (temporisation de rattachement)	Type: Unsigned4
DauxJoinFldXmit (indique des champs qui sont émis)	Type: Unsigned8
DauxJoinTx (liaison(s) JoinTx)	Type: Voir Tableau 184
DauxJoinRx (liaison(s) JoinRx)	Type: Voir Tableau 184
DauxAdvRx (liaison(s) d'annonce)	Type: Voir Tableau 184 (ou null)

L'élément DauxJoinFldXmit sélectionne les paramètres de programmation de liaisons utilisés pour des éléments DauxJoinTx, DauxJoinRx, et DauxAdvRx. Les valeurs de DauxJoinFldXmit de 0..3 correspondent à la sémantique décrite dans le Tableau 184.

Le Tableau 128 montre la structure du champ informations de rattachement. Les champs DauxJoinTx, DauxJoinRx, et DauxAdvRx peuvent faire 1..4 octet(s) (ou être vides seulement dans le cas de DauxAdvRx), selon la configuration et la sélection telles que décrites dans le Tableau 184.

Tableau 128 – Structure d'informations de rattachement

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	DauxJoinBackoff				DauxJoinTimeout			
1	DauxJoinFldXmit							
1..4	DauxJoinTx							
1..4	DauxJoinRx							
0..4	DauxAdvRx (peut être absent)							

Selon les solutions de rechange choisies, il apparaît à partir du Tableau 128 que la taille totale pourrait être une valeur quelconque entre quatre octets et quatorze octets. Cependant, des solutions de rechange doivent être choisies afin que la taille totale des champs montrés dans le Tableau 128 ne dépasse pas dix octets.

Conformément au Tableau 128, les attributs comprennent:

a) DauxJoinBackoff. Etendue maximale du repli exponentiel lors du rattachement. Si une demande de rattachement ne reçoit pas de DPDU ACK/NAK en raison d'une détection d'activité de voie CCA ou d'une émission échouée, la DLE doit se replier en sélectionnant un intervalle de temps uniformément aléatoire dans la plage de 0 s à 1 s pour la première répétition de tentative, et utiliser le premier intervalle de temps de JoinTx disponible après ce temps. Puis doubler la plage de temps avec chaque répétition de tentative. La DLE

peut répéter la tentative un nombre de fois pouvant atteindre DauxJoinBackoff fois et ne doit pas répéter la tentative plus de DauxJoinBackoff fois. A ce stade, il convient que la DLE abandonne la tentative d'envoyer un message au routeur d'annonce, retourne à l'état "provisioned" et recherche une autre annonce.

b) DauxJoinTimeout. Temps garanti, en s, pour recevoir une réponse de gestionnaire de système à une demande de rattachement. Exprimé comme un exposant, à la puissance de 2. Par exemple, si DauxJoinTimeout=5, alors la DLE peut s'attendre à un achèvement dans les 2^5 s (= 32 s). A la suite d'une temporisation, il convient que la DLE abandonne la tentative de se rattacher par l'intermédiaire du routeur d'annonce, retourne à l'état "provisioned" et recherche une autre annonce.

c) DauxJoinFldXmit:

- Bits 7..6: Unsigned2, décrivant le contenu de DauxJoinTx. Voir Tableau 184. Correspondant à dlmo.Link[ ].SchedType pour la liaison number1. La plage prise en charge est 0..2.
- Bits 5..4: Unsigned2, décrivant le contenu de DauxJoinRx. Voir Tableau 184. Correspondant à dlmo.Link[ ].SchedType pour la liaison number2. La plage prise en charge est 0..2.
- Bit 3: Si Bit3=1, émettre DauxAdvRx. Si Bit3=0, DauxAdvRx est ”null” (c'est-à-dire: vide) et n'est pas émis.
- Bits 2..1: Unsigned2, décrivant le contenu de DauxAdvRx. Voir Tableau 184. La plage prise en charge est 0..2. Correspondant à dlmo.Link[ ].SchedType pour la liaison number3. Si Bit3=0, les bits 2 et 1 n'ont pas de sens et doivent être également 0.
- Le bit 0 est réservé et doit être mis à zéro.

d) DauxJoinTx. Le ou les intervalles de temps d'émission de rattachement dans chaque cycle de supertrame, correspondant à dlmo.Link[ ].Schedule pour la liaison number1. Ce sont les occasions d'émission dans lesquelles envoyer des demandes de rattachement. Les bits 7, 6 de DauxJoinFIdXmit spécifient le format de DauxJoinTx, conformément au Tableau 184.

e) DauxJoinRx. Le ou les intervalles de temps de réception de rattachement dans chaque cycle de supertrame, correspondant à dlmo.Link[ ].Schedule pour la liaison number2. Les bits 5, 4 de DauxJoinFIdXmit spécifient le format de DauxJoinRx, conformément au Tableau 184.

f) DauxAdvRx. Liaisons de réception, pour rechercher par balayage des voisins supplémentaires après le rattachement, correspondant à dlmo.Link[ ].Schedule pour la liaison number3 lorsqu'elle est fournie. Les bits 2, 1 de DauxJoinFIdXmit spécifient le format de DauxAdvRx, conformément au Tableau 184. Il est émis et a un sens seulement lorsque DauxJoinFldXmit.Bit 3=1; sinon, sa valeur est null (vide) et aucune liaison correspondante n'est créée dans le Tableau 129.

Des liaisons sont ajoutées à dlmo.Link[ ] en fonction des paramètres dans l'annonce. Les champs sont mis conformément au Tableau 129.

Tableau 129 – Valeurs par défaut pour les liaisons créées à partir d'annonces

Nom de champ	DauxJoinTx	DauxJoinRx	DauxAdvRx (lorsque DauxJoinFldXmit.Bit3 =1)
* Index	1	2	3
SuperframeIndex	1	1	1
Type-Transmit	1	0	0
Type-Receive	0	1	1
Type-Exponential Backoff	1	0	0
Type –Idle	0	0	1
Type-Discovery	0	0	0
Type-JoinResponse	0	0	0
Type-SelectiveAllowed	1	1	1
Template1	2	1	3
Template2	Null	Null	Null
NeighborType	1	0	0
Graph Type	0	0	0
SchedType	A partir de l'annonce DauxJoinFldXmit Bits 7..6	A partir de l'annonce DauxJoinFldXmit Bits 5..4	A partir de l'annonce DauxJoinFldXmit Bits 2..1
ChType	0	0	0
PriorityType	0	0	0
Neighbor	Adresse du voisin d'annonce	Null	Null
GraphID	Null	Null	Null
Schedule	A partir de l'annonce DauxJoinTx	A partir de l'annonce DauxJoinRx	A partir de l'annonce DauxAdvRx
ChOffset	Null	Null	Null
Priority	Null	Null	Null

La liaison number3, prévue pour être utilisée pour rechercher par balayage des voisins après le rattachement, est configurée comme une liaison "idle". Normalement, les liaisons inactives sont activées par l'intermédiaire d'un sous-en-tête DAUX tel que décrit en 9.3.5.4. En outre, lorsque la DL passe à l'état de rattachement, elle doit activer la liaison number3 pendant une période de temps égale à la valeur initiale de dlmo.DiscoveryAlert.Duration (60 s par défaut). Cela conduit la DL à recueillir des informations dans la table de dlmo.Candidates et ensuite à les rapporter au gestionnaire de système par l'intermédiaire de l'alerte NeighborDiscovery, à moins que la DL ne soit reconfigurée par le gestionnaire de système pendant l'intervalle pour un résultat différent.

NOTE 2 GraphType dans le Tableau 129 est mis à zéro, indiquant que la caractéristique n'est pas applicable dans ce contexte. Voir 9.4.3.7.2.

Les liaisons créées à partir de l'annonce ont également besoin d'entrées dans dlmo.Neighbor, dlmo.Graph, et dlmo.Route. Ces entrées sont automatiquement ajoutées par la DL en même temps que les liaisons, avec des valeurs telles que montrées dans le Tableau 130, le Tableau 131 et le Tableau 132 ci-dessous.

Tableau 130 – Entrée de dlmo.Neighbor créée à partir d'annonces

Nom de champ	Valeur
*** Index**	**Adresse du routeur d'annonce**
EUI64Address	Acquise en provenance du routeur d'annonce comme décrit en 9.1.10.1
GroupCode	0
ClockSource	2
ExtGrCnt	0
DiagLevel	1
LinkBacklog	0
ExtendGraph	Null
LinkBacklogIndex	Null
LinkBacklogDur	Null

NOTE 3 ExtendGraph dans le Tableau 130 est mis à null, indiquant que la caractéristique n'est pas applicable dans ce contexte. Voir 9.4.3.4.2.

Tableau 131 – Entrée de dlmo.Graph créée à partir d'annonces

Nom de champ	Valeur
*** Index**	**1**
PreferredBranch	0
NeighborCount	1
Queue	0
MaxLifetime	0
Neighbors	Adresse du routeur d'annonce

Tableau 132 – Entrée de dlmo.Route créée à partir d'annonces

Nom de champ	Valeur
*** Index**	**1**
Size	1
Alternative	3
ForwardLimit	16
Route	Une seule entrée: 0xA001 (graphe number1)
Selector	Null

NOTE 4 Les informations relatives au chemin dans le Tableau 132 sont prévues pour être utilisées comme le chemin par défaut après que la DLE a rejoint le sous-réseau D. Les messages de rattachement destinés au proxy voisin utilisent le routage de source tel que décrit en 9.3.3.6. Des échantillons d'en-têtes de DL pour des messages de rattachement sont fournis en Annexe T.

Les mises à jour de DLMO à partir des informations de rattachement de DAUX sont réalisées en deux points dans la durée de vie de la DL. En premier lieu, lorsqu'une DLE dans l'état par défaut reçoit une annonce issue d'un mini sous-réseau D de configuration (SubnetID=1), elle a besoin de mettre à jour des attributs de DLMO avec des informations de rattachement de DAUX afin de rejoindre le mini sous-réseau D. En second lieu, lorsqu'une DLE dans l'état "provisioned" rejoint un sous-réseau D cible par l'intermédiaire d'un routeur d'annonce, elle a besoin de mettre à jour le DLMO avec des informations de rattachement de DAUX afin de rejoindre le sous-réseau D cible.

Il y a divers autres temps où une DL pourrait recevoir et traiter des annonces, tels que lors de la recherche de plusieurs voisins candidats dans l'état "provisioned", lors de la recherche de voisins candidats dans l'état "joined" ou lors de la réception de mises à jour du temps de sous-réseau D dans n'importe quel état. La réception et le traitement de telles annonces peuvent déclencher une demande de rattachement seulement dans l'état par défaut ou l'état "provisioned", et les informations de rattachement de DAUX dans l'annonce sont présentées au DLMO du destinataire seulement lorsque la DLE tente de rejoindre un mini sous-réseau D à partir de l'état par défaut ou tente de rejoindre un sous-réseau D cible dans l'état "provisioned".

9.3.5.2.4.3 Sauts discrétisés, sauts lents et processus de rattachement

Une annonce transmet une forme compressée d'une définition de supertrame. DauxChRate dans l'annonce correspond exactement à ChRate dans la supertrame, ce qui permet de distinguer une supertrame à sauts lents d'une supertrame à sauts discrétisés. Si DauxChRate=1 l'annonce indique une supertrame à sauts discrétisés; lorsque DauxChRate>1, l'annonce indique une supertrame à sauts lents.

Une annonce peut indiquer une plage de liaisons dans une supertrame, conformément au Tableau 184. Cela fournit un mécanisme pour spécifier et activer un ensemble d'intervalles de temps contigus.

Les sauts discrétisés ou sauts lents spécifiés par une DPDU d'annonce sont un sous-ensemble fonctionnel des sauts discrétisés ou sauts lents spécifiés par la supertrame et les structures de données de liaison. La différence principale réside dans le fait que l'information contenue dans la DPDU d'annonce est un peu compressée. Cette identité fonctionnelle (pour le sous-ensemble) permet que les informations transmises dans une DPDU d'annonce reçue soient utilisées pour initialiser la supertrame/les structures de données de liaison dans l'appareil qui tente d'effectuer l'opération de rattachement.

La manipulation des supertrames et liaisons d'annonce est décrite en 9.1.14. Le cross-mapping des données est indiqué dans Tableau 126 et Tableau 129. La seule différence significative réside dans le fait que, à cause de la représentation compressée, la structure de la DPDU d'annonce limite la réponse en sauts lents à 255 intervalles de temps (d'environ 2,5 s) ce qui est noté dans la description de la structure de DPDU. Ce seuil supérieur de 2,5 s n'est pas considéré comme une limite significative.

L'utilisation des sauts discrétisés, des sauts lents et des sauts hybrides est décrite de 9.1.8.4.4 à 9.1.8.4.6. La Figure 74 montre que les sauts lents peuvent être combinés avec les sauts discrétisés pour donner des sauts hybrides. Dans un routeur d'annonce configuré pour un fonctionnement hybride, une annonce peut indiquer à l'appareil se rattachant d'utiliser les liaisons à sauts lents, les liaisons à sauts discrétisés ou une combinaison des deux. Par exemple, les liaisons Tx (vers le routeur d'annonce) pourraient utiliser des sauts lents tandis que les liaisons Rx (depuis le routeur d'annonce) pourraient utiliser des sauts discrétisés. Une telle configuration est raisonnable pour un routeur configuré comme hôte de balayage actif (voir 9.1.13.3), où les liaisons à sauts lents peuvent remplir la double mission de transmettre les DPDU de sollicitation et les autres DPDU Data.

Le modèle de sauts discrétisés, de sauts lents ou de sauts hybrides est défini par les attributs de supertrame dans le Tableau 174. Le mapping est représenté dans le Tableau 126. La temporisation des liaisons dans la supertrame à sauts discrétisés, à sauts lents ou à sauts hybrides est définie dans le Tableau 181 et le Tableau 184. Le Tableau 127 montre le mapping vers le Tableau 181 tout en faisant spécifiquement référence au Tableau 184.

Dans une configuration hybride (voir Figure 74), les liaisons à sauts lents peuvent être limitées à une plage particulière d'intervalles de temps. Il s'agit de l'utilisation prévue de la "plage" dans le Tableau 184. On peut également concevoir des intervalles de temps (liaisons) spécifiques dans une période à sauts lents, et non pas une plage d'intervalles, ce qui apporte davantage de souplesse pour le concepteur d'un WISN déployé actuel.

9.3.5.2.4.4 Contrôle d'intégrité

Les DPDU qui intègrent un DAUX incluent le CRC UIT-T de 16 bits selon l'IEEE (FCS) en tant que contrôle d'intégrité sur la MPDU globale, plus un DMIC pour l'authentification en tant que contrôle d'intégrité supplémentaire. Cependant, ce DMIC ne peut pas être authentifié par une DLE destinataire sans un sens de temps partagé, une clé de sécurité partagée, et la connaissance de l'adresse EUI64Address de la DLE émettrice, qui ne sont pas disponibles à toutes les DLE qui peuvent entendre par hasard la DPDU et dériver le temps à partir de son sous-en-tête DAUX.

NOTE 1 Le temps au sein du DAUX est utilisable comme sens de temps partagé pour l'authentification de la DPDU qui contient le DAUX.

La présente norme permet à une DLE de visionner et utiliser le DAUX même si elle ne peut pas authentifier la DPDU globale. Il a été jugé insuffisant de s'appuyer sur la FCS de l'IEEE 802.15.4 comme étant le seul contrôle d'intégrité pour des annonces. Pour cette raison, un contrôle d'intégrité de 16 bits supplémentaire est inclus au sein du DAUX, couvrant seulement le contenu du DAUX lui-même.

Le contrôle d'intégrité de DAUX est similaire à la somme de contrôle selon l'UDP décrite dans l'IETF RFC 768. La somme de contrôle est le complément à un de 16 bits de la somme des compléments à un des octets qui composent le sous-en-tête de DAUX, à l'exclusion du contrôle d'intégrité lui-même, bourré d'octets zéro de remplissage à la fin (s'il y a lieu) pour obtenir un multiple de deux octets. L'ordonnancement des octets est tel qu'émis. Si la somme de contrôle calculée est zéro, elle est émise sous forme de uns partout. Une somme de contrôle émise ne comportant que des zéros partout signifie que le créateur de la DPDU n'a généré aucune somme de contrôle.

L'ordre d'émission du contrôle d'intégrité est MSB, à savoir avec le premier octet reflétant les opérations binaires sur des octets de numéro impair, avec le compte d'octets débutant à 1,

dans le sous-en-tête DAUX (octets 1, 3, 5, etc.) et le deuxième octet issu d'octets de numéro pair dans le sous-en-tête DAUX (octets 2, 4, 6, etc.).

NOTE 2 L'utilisation d'une clé secrète de sous-réseau D pour des annonces permet à ces annonces d'être dignes de confiance après que la DLE a rejoint le sous-réseau D. Les annonces sont utilisables pour des surveillances périodiques de routeurs voisins, ou pour des mises à jour périodiques du temps par des DLE avec un faible facteur d'utilisation.

L'annonce fournit seulement l'adresse DL16Address du routeur d'annonce. Cependant, une adresse EUI64Address est nécessaire pour l'échange ultérieur des DPDU avec ce routeur. Comme décrit en 9.1.14.2 la DLE répondeuse doit acquérir l'adresse EUI64Address de la DLE d'annonce.

9.3.5.2.5 Configuration des annonces

NOTE Le processus de rattachement, y compris les sollicitations, les annonces et l'utilisation des informations transmises dans les annonces, est décrit en 7.4.

La temporisation des annonces est déterminée par la structure des supertrames et des liaisons de la DLE d'annonce. Toute liaison peut inclure un fanion d'annonce, qui indique que l'annonce est incluse dans le DAUX.

Une valeur d'indice dans l'attribut dlmo.AdvSuperframe (voir Tableau 141) choisit une supertrame dans dlmo.Superframe qui doit être utilisée comme référence pour construire l'annonce. La supertrame de référence est configurée par le gestionnaire de système en établissant une supertrame, qui peut être inactive, et en se référant à son indice dans l'attribut dlmo.AdvSuperframe. La supertrame de référence ne doit pas utiliser des caractéristiques qui ne peuvent pas être représentées dans le champ d'informations dans le Tableau 124.

Une valeur zéro dans dlmo.AdvSuperframe est la valeur par défaut et elle indique que l'annonce n'a pas été configurée.

Les informations de liaison sont placées dans l'attribut dlmo.AdvJoinInfo, exactement dans le format dans lequel elles sont émises dans l'annonce dans la position correspondant au Tableau 128. De cette manière, les nouvelles liaisons de DLE JoinTx, Join Rx, et AdvRx sont spécifiées.

Le gestionnaire de système configure les supertrames dans le routeur d'annonce qui concordent avec ceux spécifiés dans la DPDU d'annonce. Au moment des liaisons JoinTx, le routeur d'annonce doit être configuré avec des liaisons pour recevoir des DPDU de rattachement. Au moment des liaisons JoinRx, le routeur d'annonce doit être configuré avec des liaisons où JoinResponse=1 (voir Tableau 182).

9.3.5.3 Sous-en-tête auxiliaire de type sollicitation

9.3.5.3.1 Généralités

NOTE Le processus de rattachement, y compris les sollicitations, les annonces et l'utilisation des informations transmises dans les annonces, est décrit en 7.4.

Une sollicitation est une demande relative à une annonce devant être émise par un hôte de balayage actif dans une portée, sur la même voie que la sollicitation elle-même (voir 9.1.13.3).

Les attributs au sein d'un DAUX de sollicitation peuvent être groupés logiquement comme:

- en-tête de sollicitation; et
- ID de sous-réseau D.

La sollicitation n'a pas un sens fiable du temps et n'a pas nécessairement une clé de sécurité secrète. Par conséquent, pour permettre au destinataire de la sollicitation de décoder son DMIC, le DMIC d'une sollicitation doit être construit en utilisant une clé de sécurité de K_global et un temps TAI nominal de zéro. Cela permet un traitement cohérent et fournit un fort contrôle d'intégrité pour la DPDU. Aucun contrôle d'intégrité supplémentaire n'est inclus dans le DAUX de la sollicitation.

Un MHR d'une sollicitation (en-tête MAC selon l'IEEE) ne doit pas fournir une adresse source ou une adresse de destination, et il ne doit pas spécifier un sous-réseau D. Voir 9.1.5.

9.3.5.3.2 Champs de sollicitation

Le Tableau 133 spécifie l'en-tête de sollicitation dans le DAUX de sollicitation.

Tableau 133 – Sous-champs de l'en-tête de sollicitation

Nom de sous-champ	Codage de sous-champ
DauxType	Type: Unsigned3 1=DAUX de sollicitation
DauxSubnetInclude indique d'émettre ou de ne pas émettre le DauxSubnetID dans la sollicitation)	Type: Unsigned1
Réservé	Type: Unsigned4=0

L'en-tête de sollicitation fait 1 octet. Conformément au Tableau 133, les éléments comprennent:

- DauxType. Mis à 1 pour indiquer un DAUX de sollicitation.
- DauxSubnetInclude. Indique d'émettre ou de ne pas émettre le champ DauxSubnetID dans la sollicitation. Si DauxSubnetInclude=0, le champ DauxSubnetID n'est pas émis, et le destinataire (hôte de balayage actif) doit utiliser la valeur par défaut de DauxSubnetID=0 pour le filtrage.

Le Tableau 134 montre la structure de l'en-tête de sollicitation.

Tableau 134 – Structure de l'en-tête de sollicitation

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	DauxType=1			DauxSubnetInclude	Reserved=0			

Le Tableau 135 spécifie les autres champs dans le DAUX de sollicitation.

Tableau 135 – Champs du DAUX de sollicitation

Nom de champ	Codage de champ
DauxSubnetID (spécifie le SubnetID)	Type: Unsigned16 (LSB)

DauxSubnetID émet un ID de sous-réseau D qui peut être utilisé comme un filtre par le destinataire, en fonction de l'attribut dlmo.SolicFilter du destinataire. Lorsque DauxSubNetInclude=0, DauxSubnetID prend la valeur par défaut de 0x0000 et n'est pas émis.

Le Tableau 136 résume la structure du DAUX de sollicitation.

Tableau 136 – Structure du DAUX de sollicitation

Nombre d'octets	bits							
	7	6	5	4	3	2	1	0
1	En-tête de sollicitation (voir Tableau 134)							
0, 2	DauxSubnetID (LSB)							

9.3.5.3.3 Configuration des sollicitations

En raison d'exigences de réglementation et de sécurité, certaines applications ne peuvent pas tolérer des DLE qui émettent des DPDU alors qu'elles sont inactives ou en transit. Par conséquent, la valeur par défaut dans la présente norme n'inclut pas des sollicitations dans sa configuration. L'intention est que les DLE soient configurées avec des sollicitations, selon le cas, au moment où elles sont configurées ou ultérieurement dans le cycle de vie.

La temporisation des sollicitations est déterminée par la structure des supertrames et des liaisons de la DLE d'annonce. L'émission d'une sollicitation est déclenchée par un champ dlmo.Link[ ].Discovery mis à une valeur de 3.

Lorsqu'une sollicitation est émise, le contenu de dlmo.SolicTemplate doit être copié textuellement dans le sous-en-tête de DAUX. Si la taille de SolicTemplate est zéro, cela doit être interprété comme une erreur de configuration et la liaison doit être ignorée.

Pour prendre en charge des exigences de réglementation et de sécurité, des sollicitations peuvent être activées et désactivées sur une base synchronisée par le gestionnaire de système par l'intermédiaire des attributs dlmo.RadioSilence, dlmo.RadioSleep, et dlmo.Superframe[ ].IdleTimer.

9.3.5.4 Sous-en-tête auxiliaire de type "activate link" (activation de liaison)

9.3.5.4.1 Généralités

Le DAUX d'activation de liaison fournit un mécanisme qui permet à un initiateur de transaction d'activer des intervalles de temps inactifs pendant une courte période de temps afin de transmettre efficacement des messages en souffrance qui se sont accumulés dans une file d'attente de messages d'un initiateur de transaction. Ces liaisons inactives, lorsqu'elles sont ainsi configurées par le gestionnaire de système, sont activées par le routeur en réponse à une salve de messages circulant à travers une DL à destination d'un voisin particulier.

Le gestionnaire de système configure:

- Une liaison d'émission inactive du côté émission, adressée à un voisin particulier ou à un groupe de voisins, avec un indice et un programme de liaison particuliers.
- Une liaison de réception inactive du côté réception, avec les mêmes indice et programme de liaison.
- Un jeu de paramètres dans la table de voisins, indiquant
  - l'indice de la liaison (LinkBacklogIndex),
  - la taille de l'accumulation qu'il convient d'avoir pour déclencher l'activation de liaison (LinkBacklog) et
  - la durée de l'activation de liaison (LinkBacklogDur).

  Voir 9.4.3.4.2 pour la définition de ces paramètres.

Lorsque l'initiateur de transaction détecte qu'il y a des DPDU Data de LinkBacklog sur sa file d'attente de messages qui peuvent être transmises à l'adresse de destination de la DPDU Data, il convient que l'initiateur de transaction utilise le sous-en-tête auxiliaire d'activation de liaison pour activer la liaison de réception inactive par l'intermédiaire du sous-en-tête

auxiliaire d'activation de liaison. Lorsque l'initiateur de transaction reçoit une DPDU ACK/NAK pour la DPDU Data, cela implique que le message a été traité et que le côté réception de la liaison inactive a été activé. Il convient que l'initiateur de la transaction active alors le côté d'émission de la liaison inactive pour le nombre désigné d'occasions de communication.

La liaison d'émission activée pourrait être adressée à un groupe de voisins, mais le côté réception de la liaison activée ne se produit toutefois que sur un seul voisin. Par conséquent, il convient de ne considérer comme étant des candidates pour la liaison activée que les seules DPDU Data adressées au voisin en question.

Le DAUX d'activation de liaison fournit un indice de liaison et un certain nombre d'occasions de communication qui sont utilisés pour activer une liaison inactive par le destinataire de la DPDU Data. Le résultat en est l'activation immédiate d'une liaison inactive pour la réception, pour le nombre d'occasions de communication indiquées par DauxActivateDur. L'initiateur de transaction du message d'activation de liaison informe, par essence, le destinataire que les messages en file d'attente suivront en étant envoyés au cours d'occasions de communication (intervalles de temps) associées à une liaison de réception particulière.

L'activation des liaisons inactives est déclenchée du côté de l'initiateur de transaction par plusieurs DPDU Data placées en file d'attente qui peuvent être acheminées vers le destinataire. Voir 9.4.3.4.2 pour une description du côté émission du message d'activation de liaison (LinkBacklogIndex, LinkBacklogDur).

9.3.5.4.2 Champs

Le Tableau 137 résume le DAUX d'activation de liaison.

Tableau 137 – Champs de DAUX d'activation de liaison

Nom de champ	Codage de champ
DauxType	Type: Unsigned3 2=DAUX d'activation de liaison
Reserved (alignement d'octets)	Type: Unsigned5=0
DauxLink_ID (identificateur pour une liaison)	Type: ExtDLUint
DauxActivateDur (nombre d'occasions de communication (intervalles de temps) pour activer la liaison, occurrences de liaison)	Type: Unsigned8

La liaison est activée pour le nombre d'occurrences de la liaison, indépendamment du fait que la liaison soit utilisée ou non. Par exemple, l'occurrence de la liaison est comptée même si elle n'est pas utilisée en raison d'une liaison de plus haute priorité dans le même intervalle de temps. La période d'activation de liaison commence par le prochain intervalle de temps complet après que le DAUX d'activation de liaison a été reçu. Voir 9.4.3.4.2.

Le Tableau 138 montre la structure du champ DAUX d'activation de liaison.

Tableau 138 – Structure de DAUX d'activation de liaison

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	DauxType			Reserved=0				
1 ou 2 octets	DauxLinkID							
1t	DauxActivateDur							

9.3.5.5 Sous-en-tête auxiliaire du type "signal quality" (qualité de signal)

9.3.5.5.1 Généralités

Le DAUX de qualité de signal rapporte la qualité du signal reçu dans une DPDU ACK/NAK, pour prendre en charge l'ensemble de diagnostics de qualité de signal aller-retour. Deux octets sont rapportés, un pour l'intensité du signal (RSSI) et un pour la qualité de signal (RSQI). RSSI et RSQI sont décrits en 9.1.15.2.

9.3.5.5.2 Champs

Le Tableau 139 résume le DAUX de rapport de qualité de signal reçu.

Tableau 139 – Champs du DAUX de rapport de qualité de signal reçu

Nom de champ	Codage de champ
DauxType	Type: Unsigned3 3=DAUX de qualité de signal
Reserved (alignement d'octets)	Type: Bit5=0
DauxRSSI (RSSI)	Type: Integer8
DauxRSQI (RSQI)	Type: Unsigned8

Le Tableau 140 montre la structure du DAUX de rapport de qualité de signal reçu.

Tableau 140 – Structure du DAUX de rapport de qualité de signal reçu

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	DauxType			Reserved=0				
1	DauxRSSI							
1	DauxRSQI							

9.4 Base d'informations de gestion de DL

9.4.1 Généralités

Pour des informations relatives au traitement général des objets de gestion normalisés dans la DL, voir 9.1.11.

9.4.2 Attributs d'objet de gestion de DL

9.4.2.1 Généralités

Le Tableau 141 résume les attributs de l'objet de gestion de DL (DLMO). Les OctetString avec une taille de zéro sont dits "null" (vides).

Tableau 141 – Attributs du DLMO *(1 de 7)*

Nom du type d'objet normalisé: DL management object (DLMO)				
Identificateur du type d'objet normalisé: 124				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
ActScanHostFract	1	Comportement de la DLE comme hôte de balayage actif	Type: Unsigned8 Classification: Static Accessibilité: Read/write Valeur par défaut: 0	Voir 9.4.2.2
AdvJoinInfo	2	Informations de rattachement devant être placées dans l'annonce	Type: OctetString Classification: Static Accessibilité: Read/write Valeur par défaut: Null	Voir 9.4.2.3
AdvSuperframe	3	Référence de supertrame pour l'annonce	Type: Unsigned16 Classification: Static Accessibilité: Read/write Valeur par défaut: 0 Plage valide: 0..32 767	Voir 9.4.2.3
SubnetID	4	Identificateur du sous-réseau D que la DLE a rejoint ou tente de rejoindre	Type: Unsigned16 Classification: Dynamic Accessibilité: Read Only Valeur par défaut: 0	Voir 9.4.2.4
SolicTemplate	5	Modèle de sous-en-tête DAUX utilisé pour des sollicitations	Type: OctetString Classification: Static Accessibilité: Read/write Valeur par défaut: Null	Voir 9.4.2.5
AdvFilter	6	Filtre utilisé sur des annonces entrantes pendant la découverte de voisins	Type: OctetString Voir Tableau 142 Classification: Static Accessibilité: Read/write Valeur par défaut: Voir 9.4.2.20	Voir 9.4.2.20
SolicFilter	7	Filtre utilisé sur des sollicitations entrantes	Type: OctetString Voir Tableau 142 Classification: Static Accessibilité: Read/write Valeur par défaut: Voir 9.4.2.20	Voir 9.4.2.20
TaiTime	8	Temps TAI pour la DLE	Type: TAINetworkTime Classification: Static Accessibilité: Read Only	Unités: 2^{-16} s Voir 9.4.2.6
TaiAdjust	9	Réajuste TaiTime à un instant programmé par le gestionnaire de système	Type: OctetString Voir Tableau 143 Classification: Dynamic Accessibilité: Read/write Valeur par défaut: Null	Voir 9.4.2.21

Tableau 141 *(2 de 7)*

Nom du type d'objet normalisé: DL management object (DLMO)				
Identificateur du type d'objet normalisé: 124				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
MaxBackoffExp	10	Exposant de repli maximal pour les répétitions de tentative	Type: Unsigned8 Classification: Static Accessibilité: Read/write Valeur par défaut: 5 Plage valide: 3..8	Voir 9.4.2.7
MaxDsduSize	11	Nombre maximal d'octets qui peuvent être pris en charge dans une seule DSDU	Type: Unsigned8 Classification: Static Accessibilité: Read/write Valeur par défaut: 96 Plage valide: 76..96	Voir 9.4.2.8
MaxLifetime	12	Durée de vie maximale d'une DPDU Data	Type: Unsigned16 Classification: Static Accessibilité: Read/write Valeur par défaut: 120 (30 s) Plage valide: 8..1 920 (2 s..480 s)	Unités: 0,25 s Voir 9.4.2.9
NackBackoffDur	13	Durée de repli après avoir reçu un NAK	Type: Unsigned16 Classification: Static Accessibilité: Read/write Valeur par défaut: 60 (15 s) Plage valide: 8..1 920 (2 s..480 s)	Unités: 0,25 s Voir 9.4.2.10
LinkPriorityXmit	14	Priorité par défaut pour des liaisons d'émission	Type: Unsigned8 Classification: Static Accessibilité: Read/write Valeur par défaut: 8 Plage valide: 0..15	Voir 9.4.2.11
LinkPriorityRcv	15	Priorité par défaut pour des liaisons de réception	Type: Unsigned8 Classification: Static Accessibilité: Read/write Valeur par défaut: 0 Plage valide: 0..15	Voir 9.4.2.11
EnergyDesign	16	Capacité en énergie de la DLE telle que conçue	Type: OctetString Classification: Constant Accessibilité: Read Only Valeur par défaut: Voir 9.4.2.22	Voir 9.4.2.22

Tableau 141 *(3 de 7)*

Nom du type d'objet normalisé: DL management object (DLMO)				
Identificateur du type d'objet normalisé: 124				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
EnergyLeft	17	Energie restante pour la DLE	Type: Integer16 Classification: Dynamic Accessibilité: Read Only	Voir 9.1.17
DeviceCapability	18	Capacités d'un appareil	Type: OctetString Voir Tableau 147 Classification: Constant Accessibilité: Read Only Valeur par défaut: Voir 9.4.2.23	Voir 9.4.2.23
IdleChannels	19	Voies radio qui doivent être inactives	Type: Unsigned16 Classification: Static Accessibilité: Read/write Valeur par défaut: 0	Voir 9.4.2.12
ClockExpire	20	Dernier délai pour expiration d'horloge	Type: Unsigned16(MSB) Classification: Static Accessibilité: Read/write Valeur par défaut: Voir description	Unités: 1 s Voir 9.4.2.13
ClockStale	21	Temporisation de source d'horloge de DL	Type: Unsigned16 Classification: Static Accessibilité: Read/write Valeur par défaut: Voir description Plage valide: 5..300	Unités: 1 s Voir 9.4.2.14
RadioSilence	22	Temporisation de silence radio	Type: Unsigned32 Classification: Static Accessibilité: Read/write Valeur par défaut: 600 Plage valide: Limité à 1..600 pour le profil de silence radio; autrement 0..232-1	Unités: 1 s Voir 9.4.2.16 et 9.1.15.4
RadioSleep	23	Période de sommeil radio. Note: La radio de la DLE sera désactivée lorsque cet attribut est mis.	Type: Unsigned32 Classification: Dynamic Accessibilité: Read/write Valeur par défaut: 0	Unités: 1 s Voir 9.4.2.17

Tableau 141 *(4 de 7)*

Nom du type d'objet normalisé: DL management object (DLMO)				
Identificateur du type d'objet normalisé: 124				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
RadioTransmitPower	24	Niveau maximal de puissance d'émission de radio	Type: Integer8	Unités: dBm Voir 9.4.2.18
			Classification: Static	
			Accessibilité: Read/write	
			Valeur par défaut: Voir texte	
			Plage valide: -20..36	
CountryCode	25	Informations relatives à l'environnement de réglementation de l'appareil	Type: Unsigned16	Voir 9.4.2.19, 9.1.15.6 et Annexe V
			Classification: Static	
			Accessibilité: Read/write	
			Valeur par défaut: 0x3C00	
Candidates	26	Liste de voisins candidats découverts par la DLE	Type: OctetString Voir Tableau 151	Voir 9.4.2.24
			Classification: Dynamic	
			Accessibilité: Read/write	
			Valeur par défaut: Null	
DiscoveryAlert	27	Contrôle de l'alerte NeighborDiscovery	Type: OctetString	Voir 9.4.2.24
			Classification: Dynamic	
			Accessibilité: Read/write	
			Valeur par défaut: Voir 9.4.2.25	
			Plage valide: Voir 9.4.2.24	
SmoothFactors	28	Facteurs de lissage pour les diagnostics	Type: OctetString Voir Tableau 153	Voir 9.4.2.25
			Classification: Static	
			Accessibilité: Read/write	
			Valeur par défaut: Voir Tableau 153	
			Plage valide: Voir Tableau 153	
QueuePriority	29	Capacité de tampon de la file d'attente pour un niveau de priorité spécifié	Type: OctetString Voir Tableau 155	Voir 9.4.2.26
			Classification: Static	
			Accessibilité: Read/write	
			Valeur par défaut: N=0	

Tableau 141 *(5 de 7)*

Nom du type d'objet normalisé: DL management object (DLMO)				
Identificateur du type d'objet normalisé: 124				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Ch	30	Modèles de saut de voie	Type: OctetString (indexé) Voir Tableau 159 Classification: Static Accessibilité: Read/write Valeur par défaut: Voir 9.4.3.2 Plage valide: Voir 9.4.3.2	Voir 9.4.3.2
ChMeta	31	Métadonnées pour l'attribut Ch	Type: Metadata_attribute Classification: Static Accessibilité: Read Only	Voir 9.4.3.2 [a]
TsTemplate	32	Modèles d'intervalle de temps	Type: OctetString (indexé) Voir Tableau 161 et Tableau 163 Classification: Static Accessibilité: Read/write Valeur par défaut: Voir 9.4.3.3 Plage valide: Voir 9.4.3.3	Voir 9.4.3.3
TsTemplateMeta	33	Métadonnées pour l'attribut TsTemplate	Type: Metadata_attribute Classification: Static Accessibilité: Read Only	Voir 9.4.3.3 et a)
Neighbor	34	Voisins	Type: OctetString (indexé) Voir Tableau 168 Classification: Static Accessibilité: Read/write Valeur par défaut: Vide Plage valide: Voir 9.4.3.4	Voir 9.4.3.4
NeighborDiagReset	35	Utilisé pour mettre à jour le champ DiagLevel au sein de l'attribut Neighbor	Type: OctetString (indexé) Voir Tableau 172 Classification: Static Accessibilité: Read/write Plage valide: Voir 9.4.3.4.3	Voir 9.4.3.4.3
NeighborMeta	36	Métadonnées pour l'attribut Neighbor	Type: Metadata_attribute Classification: Static Accessibilité: Read Only	Voir 9.4.3.4 [a]

Tableau 141 *(6 de 7)*

Nom du type d'objet normalisé: DL management object (DLMO)				
Identificateur du type d'objet normalisé: 124				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Superframe	37	Supertrames; structures et activation	Type: OctetString (indexé) Voir Tableau 175 Classification: Dynamic Accessibilité: Read/write Valeur par défaut: Vide Plage valide: Voir 9.4.3.5	Voir 9.4.3.5
SuperframeIdle	38	Utilisé pour mettre à jour les champs inactifs au sein de l'attribut Superframe	Type: OctetString (indexé) Voir Tableau 177 Classification: Dynamic Accessibilité: Read/write	Voir 9.4.3.5.3
SuperframeMeta	39	Métadonnées pour l'attribut Superframe	Type: Metadata_attribute Classification: Static Accessibilité: Read Only	Voir 9.4.3.5 [a]
Courbe	40	Graphiques	Type: OctetString (indexé) Voir Tableau 178 Classification: Static Accessibilité: Read/write Valeur par défaut: Vide	Voir 9.4.3.6
GraphMeta	41	Métadonnées pour l'attribut Graph	Type: Metadata_attribute Classification: Static Accessibilité: Read Only	Voir 9.4.3.6 [a]
Liaison	42	Relations	Type: OctetString (indexé) Voir Tableau 180 Classification: Static Accessibilité: Read/write Valeur par défaut: Vide	Voir 9.4.3.7
LinkMeta	43	Métadonnées pour l'attribut Link	Type: Metadata_attribute Classification: Static Accessibilité: Read Only	Voir 9.4.3.7 [a]
Route	44	Chemins	Type: OctetString (indexé) Voir Tableau 185 Classification: Static Accessibilité: Read/write Valeur par défaut: Vide	Voir 9.4.3.8
RouteMeta	45	Métadonnées pour l'attribut Route	Type: Metadata_attribute Classification: Static Accessibilité: Read Only	Voir 9.4.3.8 [a]

Tableau 141 *(7 de 7)*

Nom du type d'objet normalisé: DL management object (DLMO)				
Identificateur du type d'objet normalisé: 124				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
NeighborDiag	46	Diagnostics de liaisons de voisins	Type: OctetString (indexé) Voir Tableau 187	Voir 9.4.3.9
			Classification: Dynamic	
			Accessibilité: Read Only	
			Valeur par défaut: Vide	
DiagMeta	47	Métadonnées pour l'attribut NeighborDiag	Type: Metadata_attribute	Voir 9.4.3.9 et a)
			Classification: Static	
			Accessibilité: Read Only	
ChannelDiag	48	Diagnostics par voie pour gestion de spectre	Type: OctetString	Voir 9.4.2.27
			Classification: Dynamic	
			Accessibilité: Read Only	
			Valeur par défaut: Voir 9.4.2.27	
AlertPolicy	49	Rapporte des diagnostics si des problèmes de connectivité sont détectés entre des rapports HRCO réguliers	Type: OctetString	Voir 9.6.1
			Classification: Static	
			Accessibilité: Read/write	
			Valeur par défaut: Voir 9.6.1	
DLTimeout	50	La DLE peut raisonnablement se réinitialiser à l'état "provisioned" si elle ne reçoit pas une mise à jour du temps dans cet intervalle de temps	Type: Unsigned16	Voir 9.4.2.15
			Classification: Static	
			Accessibilité: Read/write	
			Valeur par défaut: Voir description	
			Plage valide: > 0	

[a] Métadonnées contenant un compte du nombre d'entrées dans le tableau et la capacité (nombre total de rangées autorisées) pour le tableau; voir 6.2.6.3 pour les détails relatifs à ce type de données.

9.4.2.2 dlmo.ActScanHostFract

dlmo.ActScanHostFract configure le comportement de la DLE comme un hôte de balayage actif, tel que spécifié en 9.1.13.3. La valeur de réglage indique la fraction du temps dans laquelle il convient que la DLE réponde lorsqu'elle a reçu un sollicitation de balayage actif. La valeur par défaut de 0 indique que la DLE n'est pas configurée pour être un hôte de balayage actif.

9.4.2.3 dlmo.AdvJoinInfo et dlmo.AdvSuperframe

dlmo.AdvJoinInfo et dlmo.AdvSuperframe configurent le contenu d'un sous-en-tête DAUX d'annonce. Leur signification est décrite en 9.3.5.2.5.

9.4.2.4 dlmo.SubnetID

dlmo.SubnetID est l'identificateur pour le seul sous-réseau D que DLE utilise actuellement ou tente de rejoindre. Les SAP de gestion de DL traitent un seul sous-réseau D actif à la fois. Si un appareil donné participe simultanément à plusieurs sous-réseaux D, cela peut être modélisé comme plusieurs instances de la DLE. La valeur dlmo.SubnetID=0 ne doit jamais être utilisée comme un ID de sous-réseau D; son utilisation indique que la DLE ne participe

pas à un sous-réseau D. La valeur dlmo.SubnetID=1 doit être utilisée exclusivement pour identifier des sous-réseaux D de configuration. Le gestionnaire de système ne met pas directement le SubnetID de la DLE; celui-ci est plutôt mis par la DLE elle-même dans le processus consistant à découvrir et rejoindre le sous-réseau D. Voir 9.1.10.2.

NOTE Dans la conception selon l'IEEE 802.15.4, SubnetID est utilisable comme filtre pour des MPDU entrantes. Comme débattu en 9.1.10.2, un DMIC fournit un filtre supplémentaire, plus fort et plus fiable une fois que la DLE a rejoint le sous-réseau D.

9.4.2.5 dlmo.SolicTemplate

dlmo.SolicTemplate est un modèle pour le sous-en-tête DAUX dans une sollicitation. Lorsqu'une sollicitation est émise, les données exactes contenues dans cet OctetString (sans taille explicite en préfixe) doivent être utilisées comme étant le sous-en-tête DAUX. La valeur est nulle (taille égale à zéro) par défaut. Voir 9.3.5.3.

9.4.2.6 dlmo.TaiTime

dlmo.TaiTime, lorsqu'il est lu par le DMAP, est rapporté comme étant la meilleure estimation de la DLE pour le temps de DL à l'instant en question. Voir 12.22.4.2 pour le codage de TAINetworkTime.

NOTE L'attribut dlmo.TaiTime est décrit comme étant en lecture seule, le temps étant acquis par la DLE auprès de ses voisins et fourni comme un service aux autres couches par l'intermédiaire du DMAP. Ce style de spécification n'est pas censé interdire les mises en œuvre, telles que celles sur des DLE qui sont des horloges-mères, où le temps est fourni à la DLE à partir d'une source alternative.

9.4.2.7 dlmo.MaxBackoffExp

dlmo.MaxBackoffExp est l'exposant de repli maximal pour les répétitions de tentative; voir 9.1.8.2 pour un débat relatif au repli exponentiel.

9.4.2.8 dlmo.MaxDsduSize

dlmo.MaxDsduSize est le nombre maximal d'octets qui peuvent être pris en charge dans une seule DSDU. Il est utilisé par la NL pour prendre des décisions de fragmentation. Sa valeur par défaut de 96 tient compte des contraintes suivantes:

- Une seule adresse EUI64Address dans le MHR. Voir 9.3.3.2.
- Un CryptoKeyIdentifier d'un octet et un décalage de saut de voie lent dans le DMXHR. Voir 9.3.3.4.
- Un seul chemin compressé dans le DROUT (c'est-à-dire: aucun routage de source au-delà du cas d'un seul saut). Voir 9.3.3.6.
- Aucun DAUX, si bien qu'une DSDU ne peut pas être combinée avec une annonce, à l'exception du DAUX d'activation de liaison lorsque l'adressage de 16 bits est utilisé.
- Un DMIC-32, pas un DMIC-64 ni un DMIC-128.

NOTE MaxDsduSize a été calculé comme suit: 15 octets pour le MHR (voir 9.3.3.2); 1 octet pour le DHR (voir 9.3.3.3); 3 octets pour le DMXHR (voir 9.3.3.4); 0 octet pour le DAUX (voir 9.3.3.5); 2 octets pour le DROUT (voir 9.3.3.6); 4 octets pour le DADDR (voir 9.3.3.7); 4 octets pour le DMIC; et 2 octets pour la FCS. Ce total de 31 octets est soustrait de la capacité de PhSDU de 127 octets, pour arriver à un MaxDsduSize de 96 octets.

Le gestionnaire de système doit réduire la valeur de dlmo.MaxDsduSize selon les besoins si des contraintes supplémentaires s'appliquent à une configuration particulière.

9.4.2.9 dlmo.MaxLifetime

dlmo.MaxLifetime est la durée maximale, en unités de 0,25 s, pendant laquelle une DPDU Data doit être maintenue dans la file d'attente de messages d'une seule DLE avant de devoir être rejetée. Le dlmo.MaxLifetime peut être neutralisé par le dlmo.Graph[ ].MaxLifetime (voir 9.4.3.6).

9.4.2.10 dlmo.NackBackoffDur

dlmo.NackBackoffDur est la durée du repli, en unités de 0,25 s, après réception d'un NAK (voir 9.1.9.4.4).

9.4.2.11 dlmo.LinkPriorityXmit et dlmo.LinkPriorityRcv

dlmo.LinkPriorityXmit et dlmo.LinkPriorityRcv sont les priorités par défaut devant être utilisées pour sélectionner des liaisons. Si aucune priorité n'est spécifiée dans dlmo.Link[ ].Priority, utiliser ces priorités. Pour les liaisons T/R, utiliser dlmo.LinkPriorityRcv comme étant la priorité pour le côté réception de la liaison. Les priorités de liaison sont décrites fonctionnellement en 9.1.8.5.

9.4.2.12 dlmo.IdleChannels

dlmo.IdleChannels fournit une liste des voies qui doivent être inactives, comme une façon rapide pour le gestionnaire de système de bloquer l'utilisation de certaines voies sur une DLE particulière sans exiger un changement coordonné des programmes de saut de voie. Une liaison se produisant sur l'une quelconque des voies désignées comme étant inactives (valeur 1) par dlmo.ActiveChannels doit être traitée comme étant inactive. Des valeurs 1 dans dlmo.IdleChannels ne doivent pas conduire au raccourcissement de séquences de saut, mais doivent laisser intactes les séquences de saut et simplement conduire à ce que toutes les liaisons sur des voies désignées soient traitées comme étant inactives. (Le raccourcissement des séquences de saut elles-mêmes est accompli par l'intermédiaire d'un attribut différent dlmo.Superframe[ ].ChMap.) Les positions des bits 0..15 correspondent aux voies 0..15. Une valeur de bit 1 indique que les liaisons utilisant la voie doivent être traitées comme étant inactives. dlmo.IdleChannels est complémentaire de dlmo.DeviceCapability.ChannelMap; dans le fonctionnement de la DLE, les deux sont logiquement combinés comme suit, le résultat étant un jeu de voies qui sont traitées comme étant inactives par la DLE:

ActiveChannels = ((NOT dlmo.IdleChannels) AND (dlmo.DeviceCapability.ChannelMap))

9.4.2.13 dlmo.ClockExpire

L'attribut dlmo.ClockExpire est le nombre maximal de secondes pendant lesquelles la DLE peut fonctionner en toute sécurité sans mise à jour d'horloge. La valeur par défaut est (1 000 s/DeviceCapability.ClockStability) et vise à maintenir la DLE synchronisée à 1 ms près au cours du processus de rattachement et ultérieurement lors de la participation à un sous-réseau D qui fournit seulement le saut de voie discrétisé. Dans les autres cas, la valeur nécessaire se met à l'échelle de manière linéaire avec la précision d'horloge exigée la plus proche ou la plus large. Voir 9.1.9.2.2.

NOTE Un appareil qui exige l'utilisation du saut de voie discrétisé est susceptible d'avoir une durée ClockExpire supérieure à la valeur par défaut ci-dessus.

9.4.2.14 dlmo.ClockStale

L'attribut dlmo.ClockStale détermine le moment où il convient que la DLE commence à accepter des mises à jour du temps issues de sources secondaires d'horloges de DL. Par exemple, si dlmo.ClockTimeout est mis à la valeur par défaut de 45 s, il convient qu'un destinataire d'horloge de DL n'accepte pas de mises à jour d'horloge issues d'une source secondaire d'horloges de DL jusqu'à ce qu'elle n'ait pas reçu une mise à jour d'horloge issue d'une source primaire d'horloges de DL pendant au moins 45 s. La valeur par défaut est 0,5 × ClockExpire. Voir 9.1.9.2.3 pour plus d'informations.

9.4.2.15 dlmo.ClockTimeout

L'attribut dlmo.ClockTimeout est le nombre maximal de secondes pendant lesquelles la DLE peut fonctionner raisonnablement dans un sous-réseau D avant de se réinitialiser à l'état configuré. La valeur par défaut est 2,0 × ClockExpire. Voir 9.1.9.2.2.

9.4.2.16 dlmo.RadioSilence

L'attribut dlmo.RadioSilence désigne le moment où la DLE doit désactiver son émetteur après avoir perdu sa connexion de sous-réseau D. Voir 9.1.15.4 pour plus d'informations.

9.4.2.17 dlmo.RadioSleep

L'attribut dlmo.RadioSleep est utilisé pour désactiver la radio de la DLE pendant une période de temps. Voir 9.1.15.4 pour plus d'informations. L'activation de cet attribut doit être légèrement retardée pour permettre la prise en charge de l'émission d'un acquittement de couche application de la TPDU du DMAP qui a fait positionner l'attribut.

9.4.2.18 dlmo.RadioTransmitPower

L'attribut dlmo.RadioTransmitPower est utilisé pour commander le niveau de puissance d'émission radio de la DLE, en dBm EIRP. Il prend comme valeur par défaut le niveau de puissance d'émission maximal autorisé de l'appareil dans le cadre d'un régime de réglementation spécifié par dlmo.CountryCode (9.1.15.6), et il est rapporté au cours du processus de rattachement par l'intermédiaire de dlmo.DeviceCapability.RadioTransmitPower. Voir 9.1.15.5.

9.4.2.19 dlmo.CountryCode

L'attribut dlmo.CountryCode fournit les contraintes sur un appareil selon le régime de réglementation applicable. Lorsqu'il est mis au cours de la configuration de la DLE, l'utilisation de la fonctionnalité de verrouillage de contenu prise en charge peut contraindre le fonctionnement de l'appareil jusqu'à la prochaine fois où il sera configuré (par exemple, éventuellement après réparation et déploiement vers une juridiction de réglementation différente). Voir 9.1.15.6 et Annexe V.

9.4.2.20 Filtres de sous-réseau

Un attribut de filtre de sous-réseau D est une chaîne de 4 octets qui indique comment une DLE doit filtrer les annonces entrantes ou les sollicitations entrantes. L'attribut AdvFilter est utilisé pour filtrer les annonces entrantes, et SolicFilter est utilisé pour filtrer les sollicitations entrantes.

AdvFilter et SolicFilter comportent chacun deux champs, un champ BitMask de 16 bits et un champ TargetID de 16 bits. Le Tableau 142 montre la structure de chaque filtre de sous-réseau D.

Tableau 142 – Octets de filtre de sous-réseau D

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
2	BitMask							
2	TargetID							

Contrairement à la plupart des attributs DLMO, les filtres de sous-réseau D utilisent des conventions d'ordonnancement d'octet LSB. Cela reflète leur utilisation, qui est d'accomplir des opération de comparaison de bits avec les éléments d'en-tête de DPDU qui sont émis dans l'ordre LSB.

Lorsqu'une DLE reçoit une annonce, elle doit vérifier le SubnetID de la DPDU entrante. L'annonce doit être ignorée, sauf si:

(DPDU.SubnetID AND AdvFilter.BitMask) égal (AdvFilter.TargetID AND AdvFilter.BitMask)

AdvFilter.BitMask doit prendre la valeur par défaut 0xFFFF, et AdvFilter.TargetID doit prendre par défaut la valeur 0x0001, le résultat étant qu'une DLE non configurée qui est dans l'état par défaut filtrera toutes les annonces à l'exception de celles reçues provenant d'un sous-réseau D de configuration avec SubnetID=1.

Lorsqu'une DLE reçoit une sollicitation, elle doit vérifier le SubnetID de la DPDU entrante. La sollicitation doit être ignorée, sauf si:

(DPDU.DauxSubnetID AND SolicFilter.BitMask) ==
(SolicFilter.TargetID AND SolicFilter.BitMask)

SolicFilter.BitMask doit prendre la valeur par défaut 0x0000, le résultat étant que les sollicitations ne sont pas filtrées par défaut.

9.4.2.21 Réajustements du temps

L'attribut dlmo.TaiAdjust inclut des champs qui sont utilisés pour réajuster dlmo.TaiTime à un instant qui est programmé par le gestionnaire de système. Cet attribut est normalement "null" (c'est-à-dire vide), sauf si une correction de temps est en cours. Son utilisation est décrite en 9.1.9.3.6. L'OctetString comporte une série de champs qui sont décrits dans le Tableau 143.

Tableau 143 – Champs de l'OctetString dlmo.TaiAdjust

Nom de champ	Codage de champ
TaiCorrection (indique l'amplitude et le sens d'une correction d'horloge TAI)	Type: Integer32 Unités: 2^{-15} s
TaiTimeToApply (indique le temps auquel la correction doit être appliquée)	Type: TAITimeRounded Unités: 1 s

NOTE L'unité TaiCorrection de 2^{-15} s a été retenue pour correspondre aux verres "de montre" très faible puissance et très précis de 32 kiHz qui sont généralement utilisés pour le matériel d'horloge continu des appareils WISN.

Le Tableau 144 montre la structure de l'OctetString.

Tableau 144 – Structure de l'OctetString dlmo.TaiAdjust

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
4	TaiCorrection							
4	TaiTimeToApply							

9.4.2.22 Capacité en énergie de la DLE

L'attribut dlmo.EnergyDesign, tel que montré dans le Tableau 145, comporte divers éléments qui indiquent la capacité en énergie de l'appareil. Les champs au sein de cet attribut sont décrits en 9.1.17.

Tableau 145 – Champs de l'OctetString dlmo.EnergyDesign

Nom de champ	Codage de champ
EnergyLife (durée de vie d'énergie de la DLE par conception; positive pour les mois, négative pour les jours)	Type: Integer16 (constant)
ListenRate (capacité en énergie de la DLE pour actionner son récepteur, en secondes par heure)	Type: ExtDLUint (constant)
TransmitRate (capacité en énergie de la DLE pour émettre des DPDU, en DPDU par minute)	Type: ExtDLUint (constant)
AdvRate (capacité en énergie de la DLE pour émettre des annonces, en DPDU par minute)	Type: ExtDLUint (constant)

Le Tableau 146 montre la structure de l'OctetString.

Tableau 146 – Structure de l'OctetString dlmo.EnergyDesign

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
2	EnergyLife							
1..2	ListenRate							
1..2	TransmitRate							
1..2	AdvRate							

9.4.2.23 Capacités de l'appareil de DLMO

L'attribut dlmo.DeviceCapability inclut divers éléments qui indiquent les capacités de l'appareil. Il s'agit d'un attribut en lecture seule, dont la plupart des valeurs des composants ne changent pas en fonctionnement normal. Elles peuvent être changées en raison de la gestion de système à distance, y compris par le téléchargement d'un nouveau firmware. L'attribut dlmo.DeviceCapability doit être rapporté au gestionnaire de système comme partie intégrante du processus de rattachement.

L'OctetString comporte une série de champs qui sont décrits dans le Tableau 147. Certains de ces champs, énumérés comme étant "static" (statiques), ne changent pas lors du fonctionnement et sont rapportés uniquement au rattachement. D'autres champs, énumérés comme étant "dynamic" (dynamiques), peuvent changer lors du fonctionnement et sont également disponibles après le rattachement par l'intermédiaire des attributs DLMO portant le même nom.

Tableau 147 – Champs de l'OctetString dlmo.DeviceCapability

Nom de champ	Codage de champ
QueueCapacity (capacité de la file d'attente qui est disponible pour des opérations de transmission)	Type: ExtDLUint (constant)
ClockStability (stabilité d'horloge nominale de cet appareil, sous la forme d'un multiple de 1×10^{-6})	Type: Unsigned8 (constant)
ChannelMap (carte des voies radio prises en charge par l'appareil)	Type: Unsigned16 (constant)
DLERoles (rôles de DLE pris en charge par la DLE)	Type: BooleanArray8 (constant)
EnergyDesign (copie de l'attribut dlmo.EnergyDesign)	Type: OctetString (constant)
EnergyLeft (copie de l'attribut dlmo.EnergyLeft)	Type: Integer16
Ack_Turnaround (voir Tableau 161) [a]	Type: ExtDLUint (constant)
NeighborDiagCapacity (capacité mémoire pour dlmo.NeighborDiag)	Type: ExtDLUint (constant)
RadioTransmitPower (copie de l'attribut RadioTransmitPower, voir 9.1.15.5)	Type: Integer8
SupportedCCAmodes (description de carte de bits des modes CCA pris en charge par l'appareil)	Type: BooleanArray8 (constant)
ConstructionOptions (à savoir les caractéristiques facultatives prises en charge par la DLE)	Type: BooleanArray8 (constant)

[a] Cela correspond à la plus grande durée exigée pour basculer du mode émission/réception au mode réception/émission, selon le mode qui survient (dans différentes DLE) à l'issue d'une DPDU Data avant la DPDU ACK/NAK suivante.

Le Tableau 148 montre la structure de l'OctetString. L'attribut EnergyDesign comporte un champ de taille d'un octet, ajouté en préfixe au début de l'OctetString de longueur non spécifiée.

Tableau 148 – Structure de l'OctetString dlmo.DeviceCapability

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1..2	QueueCapacity							
1	ClockStability							
2	ChannelMap							
1	DLERoles							
6..9	EnergyDesign (taille de l'OctetString, plus un contenu de 5..8 octets)							
2	EnergyLeft							
1..2	AckTurnaround							
1..2	NeighborDiagCapacity							
1	RadioTransmitPower							
1	SupportedCCAmodes							
1	ConstructionOptions							

Les champs comprennent:

- dlmo.DeviceCapability.QueueCapacity est le nombre de tampons dans la file d'attente qui sont disponibles pour des opérations de transmission. Cette capacité ne doit pas inclure de tampons internes qui peuvent être utilisés pour les messages qui circulent à travers la NLE. Cette valeur doit être basée sur le cas le plus défavorable, où toutes les DPDU sont de taille maximale. Lors du fonctionnement, la capacité réelle de file d'attente peut être plus grande que cette valeur rapportée. Voir 9.1.8.5.
- L'attribut dlmo.DeviceCapability.ClockStability est la stabilité nominale d'horloge à court terme de l'appareil, sous forme d'un multiple de 1×10^{-6}, en l'absence d'une correction de temps issue du sous-réseau D. Voir 9.1.9.2.2.

- L'attribut dlmo.DeviceCapability.ChannelMap est une liste de voies que l'appareil peut légalement prendre en charge dans le domaine de réglementation de l'appareil (tel que déterminé par dlmo.CountryCode, 9.1.15.6), où une valeur de zéro indique que l'appareil n'est pas autorisé à utiliser la voie. Les positions de bits 0..15 correspondent aux voies 0..15 de la présente norme (qui correspondent à leur tour aux voies DSSS 2,4 GHz 11..26 de l'IEEE 802.15.4). Si la DLE est configurée avec des liaisons qui se réfèrent à de tels voies bloquées, la DLE doit traiter ces liaisons comme étant inactives. Voir 9.1.7.2.3, 9.1.15.6, 9.4.2.19 et Annexe V.
- L'attribut dlmo.DeviceCapability.DLERoles énumère les profils de rôles de DL pris en charge par la DLE, où une valeur true indique que la DLE prend en charge le profil de rôles de DL. Voir 9.1.16 pour un débat relatif aux rôles de DLE.
  - L'Index 0 indique si la DLE prend en charge le profil de rôle d'E/S.
  - L'Index 1 indique si la DLE prend en charge le profil de rôle de routeur.
  - L'Index 2 indique si la DLE prend en charge le profil de rôle de routeur dorsal.
  - L'Index 3 indique si la DLE prend en charge le profil de rôle de silence radio. Voir 9.1.15.4.
  - Les indices 4 à 7 sont réservés et doivent être mis à la valeur false.
- Les attributs dlmo.DeviceCapability.EnergyDesign et EnergyLeft sont décrits en 9.1.17 et en 9.4.2.22.
- L'attribut dlmo.DeviceCapability.DAckTurnaround indique le temps nécessaire à la DLE pour traiter une DPDU Data reçue et répondre avec une DPDU ACK ou NAK, en unités de 2^{-15} s. Toutes les DLE doivent être capables d'utiliser les modèles d'intervalles de temps par défaut (voir Tableau 165, Tableau 166, et Tableau 167).

  La valeur DAckTurnaround est une borne supérieure sur l'intervalle de temps mesuré extérieurement exigé par l'appareil pour répondre à un signal issu de son PHY associé indiquant que la réception de DPDU est achevée et pour initier une émission PHY concernant la DPDU ACK/NAK située immédiatement après. Cette mesure implique la notation du temps lorsque le dernier symbole d'une PhPDU correspondant à la DPDU initiale d'une transaction est présenté à l'appareil de réception et que le premier signal PhPDU issu de cet appareil est détecté, où le modèle de transaction utilisé est un modèle de réception en monodiffusion utilisant la temporisation de DPDU ACK/NAK référencée à la fin de la DPDU Data venant d'être reçue.
- L'attribut dlmo.DeviceCapability.NeighborDiagCapacity indique la capacité, en octets, de l'attribut de NeighborDiag. Seuls les octets montrés dans le Tableau 187 sont inclus dans NeighborDiagCapacity. Le gestionnaire de système crée indirectement des OctetString dans NeighborDiag en positionnant les champs DiagLevel dans l'attribut Neighbor. Il convient que le gestionnaire de système ne configure pas une DLE pour remplir NeighborDiag au-delà de sa capacité énoncée, et une DLE peut échouer à accumuler des données si le gestionnaire de système excède cette capacité énoncée. Une valeur de 0x7FFF indique que la capacité est suffisante pour recueillir tous les diagnostics disponibles pour chaque dlmo.Neighbor.
- L'attribut dlmo.DeviceCapability.RadioTransmitPower est le niveau maximal de puissance pris en charge de la DLE, en dBm p.i.r.e., que la DLE peut légalement prendre en charge dans le domaine de réglementation de la DLE, tel que déterminé par dlmo.CountryCode. Voir 9.1.15.5, 9.1.15.6, 9.4.2.19 et Annexe V.
- L'attribut dlmo.DeviceCapability.SupportedCCAmodes est une liste des modes CCA que l'appareil prend en charge (voir 9.1.9.4.3):
  - Le bit 0 (Index 0) indique que le Mode CCA 1 est pris en charge.
  - Le bit 1 (Index 1) indique que le Mode CCA 2 est pris en charge.
  - Le bit 2 (Index 2) indique que le Mode CCA 3 est pris en charge.
  - Le bit 3 (Index 3) indique que le Mode CCA 4 est pris en charge.
  - Les bits 4..7 (Indices 4 à 7) sont réservés et doivent être mis à la valeur false.

- L'attribut dlmo.DeviceCapability.ConstructionOptions indique des caractéristiques facultatives que l'appareil prend en charge par construction.
  - Le bit 0 (Index 0) indique si les codes de groupe sont pris en charge dans dlmo.Neighbor.
  - Le bit 1 (Index 1) indique si les extensions de graphe sont prises en charge dans dlmo.Neighbor.
  - Le bit 2 (Index 2) indique si l'appareil est capable de recevoir des DPDU ACK/NAK en duodiffusion ou en N-diffusion.
  - Le bit 3 (Index 3) indique si l'appareil est capable de prendre en charge dlmo.Superframe.SfType=1 qui peut être nécessaire dans certaines régions pour la conformité à la réglementation.
  - Le bit 4 (Index 4) indique si l'appareil est capable de prendre en charge le champ de dlmo.Graph.Queue qui, lorsqu'il est mis à une valeur autre que zéro, réserve la capacité de tampon de file d'attente.

    Il convient que cette capacité de file d'attente ne soit pas réservée ainsi, sauf si dlmo.DeviceCapability.QueueCapacity excède l'exigence maximale pour un profil de rôle d'une DLE (voir Tableau B.8).
  - Les bits 5..7 (indices 5 à 7) sont réservés et doivent être mis à la valeur false.

9.4.2.24 Voisins candidats

L'attribut dlmo.Candidates est utilisé pour fournir au gestionnaire de système une liste de voisins candidats. La DLE peuple en toute autonomie cet attribut alors qu'il reçoit des annonces issues d'un certain nombre de voisins candidats. Cet attribut est alors transmis au gestionnaire de système afin que des décisions relatives au routage puissent être prises. Le gestionnaire de système peut réinitialiser dlmo.Candidates.N=0, signalant ainsi à la DLE d'effacer son historique d'annonces reçues et reprendre le processus de découverte de voisins.

L'attribut dlmo.DiscoveryAlert (Tableau 149) fournit au gestionnaire de système le contrôle sur la découverte de voisins et les rapports. Le gestionnaire de système met dlmo.DiscoveryAlert, et reçoit plus tard une copie de l'attribut dlmo.Candidates par l'intermédiaire de l'alerte dlmo.NeighborDiscovery. En variante, le gestionnaire de système peut lire l'attribut dlmo.Candidates sur son propre programme, ou s'arranger pour le rapporter périodiquement par l'intermédiaire du HRCO.

Tableau 149 – Champs de dlmo.DiscoveryAlert

Nom de champ	Codage de champ
Descriptor	Type: Alert report descriptor (voir Tableau 269) Par défaut: [false, 0]
Durée	Type: Unsigned16 Unités: 1 s Par défaut: 60

Le Tableau 150 montre la structure de DiscoveryAlert.

Tableau 150 – Structure de dlmo.DiscoveryAlert

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	Alert report disabled							
1	Alert report priority							
2	Durée							

Lorsque dlmo.DiscoveryAlert est activé (Descriptor.Disabled=False), la DLE doit rapporter le contenu de dlmo.Candidates.Duration des secondes plus tard en utilisant l'alerte dlmo.NeighborDiscovery. Une fois que la DLE parachève l'alerte, la DLE remet dlmo.DiscoveryAlert à zéro.

Lorsque l'alerte NeighborDiscovery est déclenchée par l'état de Duration (dlmo.DiscoveryAlert.Duration), la DLE doit automatiquement mettre le champ Duration à zéro, ce qui donne une seule alerte NeighborDiscovery chaque fois que le champ Duration est mis à une valeur autre que zéro.

L'attribut dlmo.DiscoveryAlert doit être activé et prendre par défaut la valeur Duration=60 lorsque la DLE entre dans l'état rattaché. Cette valeur par défaut indique que la DLE doit, quand elle entre dans l'état rattaché, accumuler les informations issues des annonces dans dlmo.Candidates pendant une période de 60 s, et ensuite rapporter les résultats en utilisant l'alerte dlmo.NeighborDiscovery. Voir 9.1.14.6. Le gestionnaire de système peut neutraliser cette valeur par défaut en écrivant dans dlmo.DiscoveryAlert conjointement avec la réponse de rattachement.

Les annonces issues de routeurs voisins incluent les liaisons qui peuvent être utilisées pour communiquer avec le routeur en question. Lorsque DiscoveryAlert est activé, la DLE peut utiliser ces liaisons pour interroger, avec une DSDU Data transportant une DSDU nulle, chaque routeur candidat, sur plusieurs voies, et recevoir la métrique de qualité de signal dans le DAUX. Ces informations permettent à la DEL d'élaborer une image plus exacte de la qualité de signal dans dlmo.Candidates.

Lorsque dlmo.DiscoveryAlert est désactivé, il convient néanmoins que la DLE dresse passivement une liste de dlmo.Candidates comme un processus d'arrière-plan après qu'elle a rejoint le sous-réseau D, en se basant sur des annonces de quelque nature que ce soit qu'elle reçoit au cours de l'actionnement du diagramme d'états de la DLE.

S'il y a une alerte dlmo.DL_Connectivity et DiscoveryAlert est activé, la DLE doit également envoyer une alerte dlmo.NeighborDiscovery au même moment.

Lorsque dlmo.DiscoveryAlert.Duration est mis à 0, la DLE ne doit pas envoyer d'alertes dlmo.NeighborDiscovery sur une base synchronisée. Il convient que la DLE continue de maintenir l'attribut Candidates afin qu'il puisse être lu, le cas échéant, par le gestionnaire de système en lisant directement l'attribut dlmo.Candidates directement ou en configurant la DLE pour le rapporter périodiquement par l'intermédiaire du HRCO.

Le processus de recherche par balayage des annonces est décrit en 9.1.13.

L'attribut dlmo.Candidates comporte une série de champs qui sont décrits dans le Tableau 151. Par essence, le tuple <Candidate>, <RSQI> est répété N fois, fournissant une indication de qualité de signal à plusieurs voisins.

L'attribut dlmo.Candidates peut raisonnablement exclure des entrées courantes dans la table dlmo.Neighbor, lorsque la même information pour ces voisins est disponible par l'intermédiaire du diagnostic de voisin.

Tableau 151 – Champs de l'OctetString dlmo.Candidates

Nom de champ	Codage de champ
N (nombre des voisins découverts)	Type: Unsigned8
$Neighbor_1$ (adresse 16 bits du premier candidat)	Type: ExtDLUint
$RSSI_1$ (force de signal radio du premier candidat)	Type: Integer8
$RSQI_1$ (qualité de signal radio du premier candidat)	Type: Unsigned8
…	…
$Neighbor_N$ (adresse 16 bits du $N^{ème}$ candidat)	Type: ExtDLUint
$RSSI_1$ (force de signal radio du $N^{ème}$ candidat)	Type: Integer8
$RSQI_N$ (qualité de signal radio du $N^{ème}$ candidat)	Type: Unsigned8

Le Tableau 152 montre la structure de Candidates.

Tableau 152 – Structure de dlmo.Candidates

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	N							
1..2	$Neighbor_1$							
1	$RSSI_1$							
1	$RSQI_1$							
…	…							
1..2	$Neighbor_N$							
1	$RSSI_N$							
1	$RSQI_N$							

Les champs comprennent:

- dlmo.Candidates.N est le nombre de voisins qui ont été découverts. Une DLE doit prendre en charge au moins cinq (5) entrées candidates. Si plusieurs voisins sont découverts, la DLE peut rapporter seulement les meilleurs candidats sur la base de la qualité de la liaison radio.
- dlmo.Candidates.$Neighbor_N$ est l'adresse de 16 bits de chaque voisin candidat dans le sous-réseau D.
- dlmo.$RSSI_N$ Indique l'intensité du signal radio en dBm provenant de chaque voisin candidat, sur la base des annonces reçues et possiblement d'autres DPDU. Voir 9.1.15.2 pour une description de l'attribut RSSI, y compris le biais de dBm fixe dans les valeurs de mesure reportées.
- dlmo.$RSQI_N$ indique la qualité du signal radio provenant de chaque voisin candidat, sur la base des annonces reçues et possiblement d'autres considérations. Un nombre plus élevé indique un signal radio meilleur. Voir 9.1.15.2 pour une description de l'attribut RSQI.

9.4.2.25 Facteurs de lissage

L'attribut dlmo.SmoothFactors fournit les facteurs de lissage pour l'attribut dlmo.NeighborDiag. L'utilisation de ces facteurs est décrite en 9.1.15.3.

Trois champs dans dlmo.NeighborDiag impliquent le lissage exponentiel: RSSI, RSQI, et ClockBias. Le facteur de lissage α (alpha) pour chacune de ces valeurs est configurable individuellement. Alpha est exprimé en pourcentage entier dans la plage 0..100. RSSI et RSQI prennent par défaut la valeur de 10 %, si bien que les valeurs tendent à refléter l'historique récent. ClockBias prend par défaut la valeur de 1 %, car cette valeur est destinée à montrer le biais d'horloge sur une période de temps prolongée.

Chaque facteur de lissage implique trois valeurs: x_AlphaHigh, x_AlphaLow et x_Threshold pour différents x. Si les nouvelles données se situent en dessous du seuil, utiliser x_AlphaLow comme étant le facteur de lissage; autrement, utiliser x_AlphaHigh.

Les champs dans dlmo.SmoothFactors sont décrits dans le Tableau 153.

Tableau 153 – Champs de l'OctetString dlmo.SmoothFactors

Nom de champ	Codage de champ	Par défaut
RSSI_Threshold (seuil pour RSSI)	Type: Integer16	0
RSSI_AlphaLow (AlphaLow pour RSSI)	Type: Unsigned8	10
RSSI_AlphaHigh (AlphaHigh pour RSSI)	Type: Unsigned8	10
RSQI_Threshold (seuil pour RSQI)	Type: Integer16	0
RSQI_AlphaLow (AlphaLow pour RSQI)	Type: Unsigned8	10
RSQI_AlphaHigh (AlphaHigh pour RSQI)	Type: Unsigned8	10
ClockBias_Threshold (seuil pour ClockBias)	Type: Integer16	0
ClockBias_AlphaLow (AlphaLow pour ClockBias)	Type: Unsigned8	1
ClockBias_AlphaHigh (AlphaHigh pour ClockBias)	Type: Unsigned8	1

Le Tableau 154 montre la structure de SmoothFactors.

Tableau 154 – Structure de dlmo.SmoothFactors

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
2	RSSI_Threshold							
1	RSSI_AlphaLow							
1	RSSI_AlphaHigh							
2	RSQI_Threshold							
1	RSQI_AlphaLow							
1	RSQI_AlphaHigh							
2	ClockBias_Threshold							
1	ClockBias_AlphaLow							
1	ClockBias_AlphaHigh							

9.4.2.26 dlmo.QueuePriority

9.4.2.26.1 Généralités

dlmo.QueuePriority est un attribut qui permet au gestionnaire de système de spécifier la capacité de tampon nominale dans la file d'attente de transmission de la DLE pour les niveaux spécifiques de priorité. Comme décrit en 9.1.8.5, le gestionnaire de système peut configurer la capacité de tampon nominale de la DLE pour limiter le nombre des tampons qui peuvent être utilisés pour transmettre des DPDU Data de basse priorité. Par exemple, le gestionnaire de système peut configurer dlmo.QueuePriority si bien que pas plus de 3 tampons ne doivent être utilisés pour transmettre des DPDU Data avec une priorité ≤ 2.

La capacité nominale de la file d'attente de transmission peut être trouvée dans dlmo.DeviceCapability.QueueCapacity (voir 9.4.2.23).

9.4.2.26.2 Sémantique

Le Tableau 155 spécifie les champs pour dlmo.QueuePriority.

Tableau 155 – Champs de dlmo.QueuePriority

Nom de champ	Codage de champ
N (nombre des priorités spécifiées, 0..15, valeur par défaut N=0)	Type: Unsigned8
Priority1 (première priorité)	Type: Unsigned8
QMax1 (capacité de premier tampon)	Type: Unsigned8
...	
$Priority_N$ ($N^{ème}$ priorité)	Type: Unsigned8
$QMax_N$ (capacité de $N^{ème}$ tampon)	Type: Unsigned8

Par exemple, si la priorité est 2 et QMax est 3, alors pas plus de 3 tampons de file d'attente ne doivent être utilisés pour transmettre des DPDU Data avec une priorité ≤ 2. Ce compte ne doit pas inclure des DPDU Data qui utilisent la capacité de file d'attente qui a été mise de côté pour des données DPDU Data transmises le long d'un graphe particulier, en se basant sur dlmo.Graph[ ].Queue. La priorité doit être énumérée dans l'ordre croissant, si bien que PriorityX doit être inférieur à $Priority_{X+1}$. De même, $QMax_X$ doit être inférieur à $QMax_{X+1}$.

Le compte $QMax_X$ établit les intervalles de temps disponibles maximaux pour les messages de faible priorité, pas les intervalles de temps réservés pour les messages de faible priorité. En fait, $QMax_X$ assure que le reste de la file d'attente est disponible pour les DPDU ayant la priorité 1 + $Priority_X$.

Comme décrit en 9.2.2, l'indicateur DE dans un en-tête de DPDU indique si une DPDU Data dans la file d'attente est éligible à être rejetée en faveur d'une DPDU entrante ayant une plus haute priorité.

La valeur par défaut, où $N = 0$, indique que le gestionnaire de système n'a pas configuré une limite sur le nombre de tampons de transmission qui peuvent être utilisés pour des DPDU de faible priorité.

Le Tableau 156 montre la structure de dlmo.QueuePriority.

Tableau 156 – Structure de dlmo.QueuePriority

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	N							
2 (opt)	$Priority_1$							
3 (opt)	$QMax_1$							
...	...							
2N (opt)	$Priority_N$							
2N+1 (opt)	$QMax_N$							

9.4.2.27 dlmo.ChannelDiag

9.4.2.27.1 Généralités

L'attribut dlmo.ChannelDiag est un attribut dynamique en lecture seule qui rapporte les taux d'échec d'émission de DPDU sur chaque voie radio prise en charge par la norme. Cela permet au gestionnaire de système d'avoir conscience des problèmes de connectivité cohérente sur une base voie par voie, comme un diagnostic pour la gestion de spectre dans un sous-réseau D.

Deux diagnostics sont rapportés pour chaque voie, chacun indiquant un type différent d'échec. $NoACK_N$ indique le pourcentage du temps pendant lequel les émissions de DPDU en monodiffusion sur la voie N n'ont pas entraîné la réception d'une DPDU ACK/NAK. $CCABackoff_N$ indique le pourcentage de temps pendant lequel l'appareil a abandonné une émission initiatrice de transaction sur la voie N en raison de la CCA.

Dans un certain nombre de situations, le repli de CCA est une partie intégrante du fonctionnement normal du sous-réseau D et n'est pas l'indicateur d'une qualité médiocre de voie. En particulier, un conflit pour des liaisons partagées conduira au repli de la CCA. Par conséquent, la DLE peut être sélective dans son compte de replis de CCA. Il est recommandé qu'il convient que les replis de CCA ne soient pas comptés normalement s'ils se produisent dans des liaisons partagées au sein de supertrames de saut de voie discrétisé. Les liaisons partagées sont décrites en 9.1.8.2.

Les valeurs de ChannelDiag sont des entiers signés de 8 bits.

- Une valeur de 0 indique qu'aucune émission n'a été tentée sur la voie.
- Les valeurs positives dans la plage 1..101 indiquent le taux d'échec en pourcentage plus un. Par exemple, une valeur de $CCABackoff_6$ indique que 25 % des émissions initiatrices de transactions sur la voie 6 ont été abandonnées en raison de la CCA.
- Les valeurs négatives dans la plage -1 à -101 indiquent le taux d'échec en pourcentage comme étant un nombre négatif moins un. Les taux d'échec sont rapportés sous forme de nombres négatifs s'ils sont basés sur 5, ou moins de 5, émissions tentées. Par exemple, une valeur $NoACK_8$ de -34 indique que 33 % des émissions en monodiffusion sur une voie particulière n'ont pas reçu d'acquittement et que 5, ou moins de 5, émissions ont été tentées sur cette voie.

La DLE peut sélectivement sauter des liaisons d'émission par anticipation d'une émission échouée, comme décrit en 9.1.7.2.4. Il convient que de telles liaisons sautées soient traitées comme l'équivalent du NAK pour la voie applicable.

Chaque fois que ChannelDiag est lu par le gestionnaire de système ou rapporté périodiquement par le HRCO, ses accumulateurs sous-jacents doivent être remis à zéro.

9.4.2.27.2 Sémantique

Le Tableau 157 spécifie les champs pour dlmo.ChannelDiag.

Tableau 157 – Champs de dlmo.ChannelDiag

Nom de champ	Codage de champ
Count (nombre des émissions en monodiffusion tentées pour toutes les voies)	Type: Unsigned16
NoACK0 (pourcentage du temps pendant lequel des émissions de temps sur la voie 0 n'ont pas reçu une DPDU ACK)	Type: Integer8
CCABackoff0 (pourcentage du temps pendant lequel des émissions sur la voie 0 ont été abandonnées en raison de la CCA)	Type: Integer8
…	…
NoACK15 (pourcentage du temps pendant lequel des émissions de temps sur la voie 15 n'ont pas reçu une DPDU ACK)	Type: Integer8
CCABackoff15 (pourcentage du temps pendant lequel des émissions sur la voie 15 ont été abandonnées en raison de la CCA)	Type: Integer8

Le Tableau 158 montre la structure de dlmo.ChannelDiag.

Tableau 158 – Structure de dlmo.ChannelDiag

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	Count							
1	$NoACK_0$							
1	$CCABackoff_0$							
...	...							
1	$NoACK_{15}$							
1	$CCABackoff_{15}$							

9.4.3 Attributs de DLMO (OctetStrings indexés)

9.4.3.1 Généralités

Les attributs d'objet de gestion d'OctetString indexés sont des ensembles structurés de données, de conception similaire à des tables SQL. Par exemple, la DLE maintient une liste de voisins dans un attribut OctetString indexé appelé dlmo.Neighbor, où chaque voisin peut être vu comme une rangée dans une table, avec la structure de chaque rangée telle que montrée dans le Tableau 168 et dans le Tableau 169. Un indice de 15 bits pour chaque rangée, correspondant dans le présent cas à l'adresse de 16 bits du voisin, est fourni pour chaque attribut OctetString indexé dans le DLMO.

Pour la cohérence du traitement, tous les attributs OctetString indexés dans la DL incluent un indice dans le premier champ qui est de type ExtDLUint. Cependant, dans le cas de dlmo.Ch, dlmo.TsTemplate, et dlmo.Superframe, l'indice est limité à une plage de 1..127, garantissant donc que l'indice peut être représenté dans un seul octet. Les références à ces structures peuvent également être compressées en un seul octet.

Pour un OctetString indexé, l'indice égal à zéro ne doit pas être utilisé dans le DLMO, sauf pour indiquer une entrée "null" (c'est-à-dire vide).

La Figure 90 montre la relation entre certains des attributs OctetString indexés de DLMO. Les relations référentielles sont montrées par des flèches. Par exemple, dlmo.Link se réfère à dlmo.TsTemplate, aussi une flèche est-elle montrée pointant dans le sens de la référence. Cela correspond grossièrement à une relation un à plusieurs, où un même modèle peut être référencé par plusieurs liaisons.

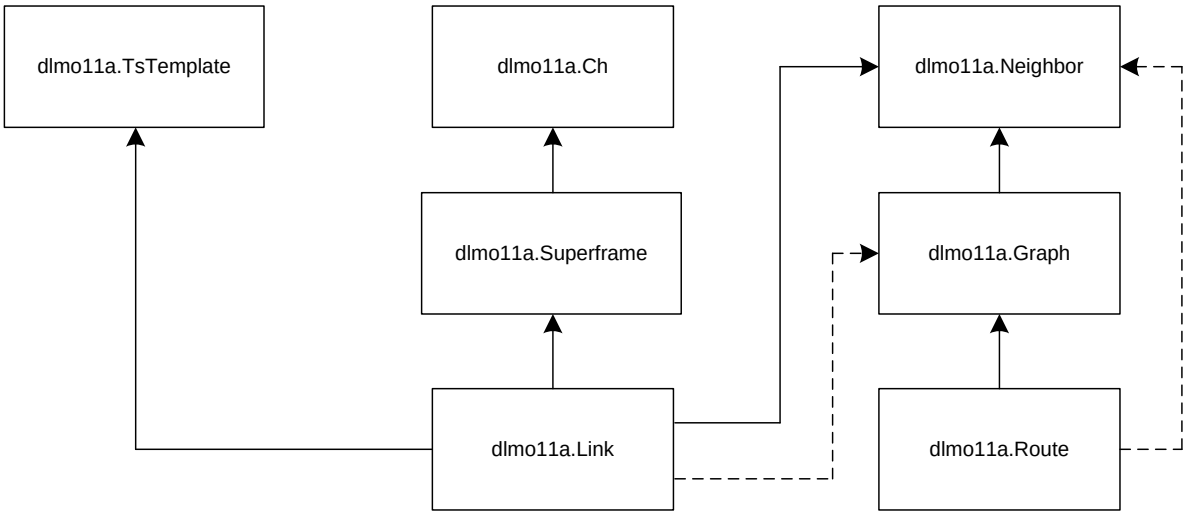


Figure 90 – Relation entre attributs indexés de DLMO

Comme montré à la Figure 90, dlmo.TsTemplate est référencé par dlmo.Link. Les modèles d'intervalle de temps spécifient la structure et la temporisation de transaction lorsqu'une liaison est utilisée.

dlmo.Link, dlmo.Superframe, et dlmo.Ch sont reliés. Les supertrames sélectionnent un modèle de saut de voie et un programme cyclique. Les liaisons décrivent les diverses actions qui sont entreprises au cours de chaque cycle de supertrame. Chaque liaison se réfère à une supertrame.

Pour les transactions en monodiffusion (et en duodiffusion), dlmo.Link se réfère à dlmo.Neighbor. Une seule et même référence peut englober un groupe de voisins. Si une liaison est réservée pour l'utilisation d'un certain graphe ou donne un accès préférentiel à un certain graphe, alors dlmo.Link se référera également à dlmo.Graph. Sachant qu'on peut configurer une liaison sans une référence à un graphe, cette référence est montrée sous la forme d'une ligne à traits pointillés à la Figure 90.

Les attributs dlmo.Route et dlmo.Graph sont reliés en ce que les chemins se réfèrent habituellement à des graphiques. Les attributs dlmo.Graph et dlmo.Neighbor sont reliés en ce que les graphes se réfèrent à des voisins. Les attributs dlmo.Route et dlmo.Neighbor sont reliés d'une manière lâche (indiquée par une ligne à traits pointillés à la Figure 90) en ce que le routage de source peut se référer implicitement à un voisin.

Une relation supplémentaire, non montrée à la Figure 90, existe entre dlmo.Neighbor et dlmo.NeighborDiag. Tous les deux sont indexés par l'adresse de 16 bits des voisins de la DLE, mais dans des buts différents.

- dlmo.Neighbor: Le gestionnaire de système fournit cet attribut OctetString indexé "static" pour permettre à la DLE de communiquer avec ses voisins immédiats.
- dlmo.NeighborDiag: Le gestionnaire de système configure cet attribut OctetString indexé "dynamic" pour permettre à la DLE de recueillir et rapporter périodiquement le diagnostic pour ses voisins immédiats.

9.4.3.2 dlmo.Ch

9.4.3.2.1 Généralités

L'attribut dlmo.Ch est un ensemble OctetString indexé contenant des modèles de saut de voie qui sont disponibles.

Les modèles de saut de voie 1 à 5 sont réservés comme valeurs par défaut normalisées, telles que décrites en 9.1.7.2.5. Des modèles complémentaires de saut de voie peuvent être ajoutés.

Chaque DLE peut stocker plusieurs modèles de saut de voie, avec un indice propre à chaque modèle. Les annonces supposent que des modèles de saut de voie pour le processus de rattachement sont configurés dans la DLE lorsque l'annonce est reçue. Donc, tout modèle de saut de voie référencé dans une annonce doit concorder avec une des valeurs par défaut.

Le gestionnaire de système insère, met à jour ou efface des modèles de saut de voie en envoyant au DMAP un modèle de saut de voie, accompagné d'un indice unique et (si sélectionné) d'un temps TAI de basculement.

Pour un modèle donné de saut de voie, la norme fournit un mapping entre les numéros de voie de DL et les numéros de voie plus généraux de la MAC selon l'IEEE 802.15.4. Tels qu'appliqués à la radio DSSS dans la bande de 2,4 GHz selon l'IEEE 802.15.4, les numéros de voie doivent être limités à la plage 0..15, correspondant aux numéros de voies 11..26 (page de voies 0) de l'IEEE 802.15.4, dans le même ordre. Les modèles de saut de voie pour cette radio ne doivent pas excéder une taille de 16.

NOTE Les types de données dlmo.Ch sont limités à la radio avec 16 voies ou moins de 16 voies. Les radios du futur avec plus de voies impliqueront de fournir la prise en charge pour une représentation moins compressée.

Cinq modèles de saut de voie par défaut sont réservés et définis par la norme. Des modèles de saut de voie par défaut sont énumérés et décrits en 9.1.7.2.5.

9.4.3.2.2 Sémantique

Le Tableau 159 spécifie les champs pour dlmo.Ch. Un indice, utilisé pour identifier chaque modèle de saut de voie, est cohérent avec les conventions de DL pour les OctetStrings indexés.

Tableau 159 – Champs de dlmo.Ch

Nom de champ	Codage de champ
* Index	Type: ExtDLUint (utilisé comme un indice) Plage valide: 1..127
Size	Type: Unsigned8 Plage valide: 1..16
Seq (modèle de saut de voie, avec des entrées Size)	Type: SEQUENCE OF Unsigned4 (SIZE (Size))

Le Tableau 160 montre la structure de dlmo.Ch.

Tableau 160 – Structure de dlmo.Ch

<table>
<tr><th rowspan="2">Nombre d'octets</th><th colspan="8">Bits</th></tr>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>1</td><td colspan="8">Index (1..127)</td></tr>
<tr><td>1</td><td colspan="8">Size</td></tr>
<tr><td rowspan="4">(Size+1)/2</td><td colspan="4">Seq_1</td><td colspan="4">Seq_0</td></tr>
<tr><td colspan="4">Seq_3</td><td colspan="4">Seq_2</td></tr>
<tr><td colspan="4">...</td><td colspan="4">...</td></tr>
<tr><td colspan="4">0x0000 lorsque Size est un nombre impair; Seq_{Size-1} lorsque Size est un nombre pair</td><td colspan="4">$Seq_{2(n-2)}$</td></tr>
</table>

9.4.3.3 dlmo.TsTemplate

9.4.3.3.1 Généralités

dlmo.TsTemplate est un ensemble OctetString indexé qui contient des modèles d'intervalle de temps. Les modèles d'intervalle de temps décrivent la temporisation de la transaction D.

Les modèles d'intervalle de temps 1, 2, et 3 doivent être réservés comme valeurs par défaut normalisées, comme énuméré au Tableau 165, au Tableau 166 et au Tableau 167. D'autres modèles d'intervalle de temps peuvent être ajoutés par le gestionnaire système.

Le gestionnaire de système insère, met à jour ou efface des modèles d'intervalle de temps en envoyant au DMAP un modèle, accompagné d'un indice unique et (si sélectionné) d'un temps TAI de basculement.

Des décalages de temps de modèle sont spécifiés en unités de 2^{-20} s, ce qui représente approximativement 1 μs. La durée d'intervalle de temps est fixée par les supertrames qui utilisent le modèle.

Les types de modèle incluent des modèles d'initiateur de réception et de transaction. Les deux types de modèle incluent des acquittements pour des transactions en monodiffusion. Les mêmes modèles peuvent également être utilisés pour des liaisons de diffusion, telles que des sollicitations, qui n'ont pas besoin d'acquittements et n'utilisent pas les temporisations de DPDU ACK/NAK.

Les modèles peuvent être définis sur trois niveaux:

- Des modèles par défaut sont définis dans la norme. Ce sont les modèles d'intervalle de temps exigés pour le rattachement (Tableau 165, Tableau 166 et Tableau 167). Le modèle d'indice=1 décrit une transaction de réception générique et est utilisé pour recevoir des réponses de rattachement. Le modèle d'indice= 2 décrit une transaction d'émission générique et est utilisé pour émettre des demandes de rattachement. Le modèle d'indice=3 décrit une transaction qui actionne son récepteur pendant un intervalle de temps entier, destiné aux opérations telles que rechercher par balayage des annonces ou recevoir des DPDU de saut de voie lent à temporisation lâche. Ces modèles génériques doivent être utilisés au cours de la configuration et du rattachement, et peuvent également être utilisés pour d'autres buts.
- Des modèles configurés peuvent être ajoutés lors du processus de configuration, avec une durée de vie jusqu'à ce que la DLE ait rejoint le sous-réseau D. Voir 9.1.14.4.
- Des modèles spécifiques à un sous-réseau peuvent être fournis à la DLE après que le processus de rattachement a été parachevé.

La temporisation des émissions de DPDU Data est basée sur l'horloge interne de l'initiateur de transaction. La temporisation des DPDU ACK/NAK est spécifiée comme étant le décalage de temps par rapport au point de référence qui est indiqué dans le modèle. Habituellement, la plage de temps configurée pour un initiateur de transaction est plus étroite que la plage de temps pour les récepteurs de transaction, afin de rendre compte des temps de garde.

Les modèles d'intervalle de temps doivent fournir une fenêtre de réception d'au moins 192 μs, ce qui implique que les DLE doivent être capables de commander la temporisation d'émission avec une exactitude d'au moins ±96 μs, à savoir ±6 périodes symboles PHY.

Par convention, la temporisation de modèle d'intervalle de temps est spécifiée en étant basée sur les temps de début et de fin des DPDU. La temporisation de PhPDU, dépendante des détails de la couche physique qui contient la DPDU, peut être inférée à partir des temps de DPDU. Le temps de début de DPDU, tel que spécifié dans des modèles d'intervalle de temps, utilise une convention selon laquelle la DPDU commence à l'instant juste avant que le premier octet dans l'en-tête de la DPDU ne soit émis. Cette convention s'applique aux émissions de DPDU Data ainsi qu'aux DPDU ACK/NAK.

NOTE Dans les mises en œuvre réelles basées sur l'IEEE 802.15.4 (2,4 GHz), les signaux de temporisation de PhL sont parfois déclenchés lorsqu'un SFD (délimiteur de début de trame) est complètement émis ou reçu. Dans de tels cas, le temps de début de la DPDU se situe à 1 octet, ou 32 μs, après le point de référence.

Le temps de réponse ACK/NAK est communément spécifié par rapport au temps de fin de la DPDU Data, la DPDU ACK/NAK se produisant approximativement 1 ms après.

En variante, les DPDU ACK/NAK peuvent être synchronisées par rapport au temps de fin programmé de l'intervalle de temps. En plaçant des DPDU ACK/NAK à un décalage fixe au sein de l'intervalle de temps, il est possible de respecter les exigences de réglementation qui interdiraient l'émission d'une DPDU ACK/NAK après une très courte DPDU Data, si l'expéditeur de la DPDU ACK/NAK avait aussi émis près de la fin de l'intervalle antérieur.

Un tel placement par rapport à la fin de l'intervalle de temps prend également en charge les routeurs qui opèrent sur plusieurs voies multiples même temps, partageant une antenne commune, ou des appareils dont les antennes sont situées à étroite proximité l'une de l'autre. Dans les deux cas, sans un tel alignement programmé, les DPDU Data et/ou DPDU ACK/NAK émises sur une même voie pourraient involontairement bloquer une réception à chevauchement de temps dont la réception est tentée sur une autre voie.

Quand les temps de DPDU ACK/NAK sont définis comme des décalages par rapport à la fin de l'intervalle de temps, les décalages de temps, qui sont des nombres entiers non signés, doivent être interprétés comme se référant à un temps avant la fin de l'intervalle de temps.

Les modèles de récepteur de transaction spécifient:

- La plage de temps où le premier octet d'une DPDU Data peut être reçu, indiquant une plage de temps pour activer et désactiver le récepteur de radio. Une plage de temps qui excède la durée de l'intervalle de temps indique que la plage se prolonge seulement jusqu'à la fin programmée de l'intervalle de temps.
- La plage de temps de retard de DPDU ACK/NAK.

Il convient qu'un répondeur de transaction pour une transaction en monodiffusion réponde par une DPDU ACK/NAK aussitôt que possible dans cette plage, à l'exception des DPDU ACK/NAK en n-diffusion volontairement échelonnées. Le temps de retard d'une DPDU ACK/NAK peut être spécifié en référence à la fin de la DPDU reçue ou comme étant un décalage arrière par rapport à la fin programmée de l'intervalle de temps.

- S'il est acceptable de fonctionner après la frontière de l'intervalle de temps une fois que la réception d'une DPDU commence, prolongeant la durée d'intervalle de temps.

Le dépassement d'une frontière d'intervalle peut être utilisé pour prendre en charge le saut de voie discrétisé, où des DLE initiatrices de transactions peuvent avoir un sens temporel biaisé des frontières d'intervalle de temps. La prise en charge de ces DLE implique d'autoriser la réception même si la transaction dépasse une frontière d'intervalle.

Les modèles d'initiateur de transaction spécifient:

- La plage de temps pour commencer l'émission. Une DLE conforme peut commencer son émission à tout moment au sein de la plage de temps, en fonction de son horloge interne de DLE. La plage de temps est basée sur le temps de début de la DPDU Data. L'opération de CCA, le cas échéant; le SHR et le PHR de la PhPDU précèdent l'événement et peuvent donc se produire avant que la plage de temps antérieure admise ait commencé l'émission.
- La plage de temps de retard de DPDU ACK/NAK. Un répondeur de transaction répondra habituellement aussitôt que possible au sein de cette plage, conformément à son modèle d'intervalle de temps mais sous réserve des contraintes de réglementation sur le retard exigé minimal depuis la dernière émission antérieure par le même appareil. Le temps de retard d'une DPDU ACK/NAK peut être spécifié en relation avec la fin de la réception de la DPDU Data ou avec la fin programmée de l'intervalle de temps.
- L'indication du mode CCA à utiliser avant l'émission permet de contrôler les émissions concurrentes ou en cours.
- L'indication de savoir si, oui ou non, il convient que la DLE continue périodiquement d'actionner son récepteur pour la plage entière de temps de retard de DPDU ACK/NAK, même après avoir reçu une DPDU ACK/NAK (destinée à la prise en charge de duodiffusion; voir 9.1.9.4.7).

9.4.3.3.2 Sémantique

Il y a deux variantes de dlmo.TsTemplate, l'une pour un modèle de récepteur de transaction et l'autre pour un modèle d'initiateur de transaction. Les variantes sont distinguées par un type de 2 bits qui est le premier élément. Type=0 indique un modèle de récepteur de transaction, alors que le type=1 indique une modèle d'initiateur de transaction. Les types 2 et 3 sont réservés pour usage futur.

Le Tableau 161 spécifie les champs pour le modèle de récepteur de transaction.

Tableau 161 – Champs du modèle de récepteur de transaction

Nom de champ	Codage de champ
* Index	Type: ExtDLUint (utilisé comme un indice) Plage valide: 1..127
Type (indique qu'il s'agit d'un modèle de récepteur de transaction)	Type: Unsigned2 Valeurs nommées: 0: modèle de récepteur de transaction 1: voir Tableau 163; les autres choix sont réservés
AckRef (indique la référence pour la plage de temps de DPDU ACK/NAK)	Type: Unsigned1: Valeurs nommées: 0: décalage par rapport à la fin de la DPDU entrante; 1: décalage négatif par rapport à la fin de l'intervalle de temps
RespectBoundary (spécifie si, oui ou non, les frontières d'intervalle de temps doivent être respectées (c'est-à-dire si, oui ou non, une transaction D peut se prolonger au-delà de la frontière d'intervalle de temps))	Type: Boolean1: false: les frontières d'intervalle de temps peuvent ne pas être respectées; true: les frontières d'intervalle de temps doivent être respectées
Reserved (alignement d'octets)	Type: Unsigned4=0
WakeupDPDU (temps le plus tôt auquel la réception de DPDU peut être censée commencer, indiquant quand activer le récepteur de radio décalage par rapport au temps de début programmé de l'intervalle de temps)	Type: Unsigned16 Unités: 2^{-20} s
TimeoutDPDU (temps le plus tard auquel la réception de DPDU peut être censée commencer, indiquant quand désactiver le récepteur si aucune DPDU entrante n'est détectée; décalage par rapport au temps de début programmé de l'intervalle de temps)	Type: Unsigned16 Unités: 2^{-20} s
AckEarliest (début de la plage de temps de retard de DPDU ACK/NAK, le début de cette PhSDU (DPDU) étant la référence de temps. La sémantique dépend du champ AckRef: si AckRef=1, soustraire la valeur à la valeur du temps de fin programmé de l'intervalle de temps pour déterminer AckEarliest)	Type: Unsigned16 Unités: 2^{-20} s
AckLatest (fin de la plage de temps de retard de DPDU ACK/NAK, le début de cette PhSDU (DPDU) étant la référence de temps. La sémantique dépend du champ AckRef: si AckRef=1, soustraire la valeur à la valeur du temps de fin programmé de l'intervalle de temps pour déterminer AckLatest)	Type: Unsigned16 Unités: 2^{-20} s

Le Tableau 162 spécifie la structure du modèle de récepteur de transaction.

Tableau 162 – Structure du modèle de récepteur de transaction

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	* Index (range 1..127)							
1	Type=0		AckRef	RespectBoundary	Réservé			
2	WakeupDPDU							
2	TimeoutDPDU							
2	AckEarliest							
2	AckLatest							

Le Tableau 163 spécifie les champs pour le modèle d'initiateur de transaction.

Tableau 163 – Champs du modèle d'initiateur de transaction

Nom de champ	Codage de champ
* Index	Type: ExtDLUint (utilisé comme un indice) Plage valide: 1..127
Type (indique qu'il s'agit d'un modèle d'initiateur de transaction)	Type: Unsigned2 Valeurs nommées: 0: voir Tableau 161; 1: modèle d'initiateur de transaction; les éléments restant sont réservés.
AckRef (indique la référence pour la plage de temps de retard de DPDU ACK/NAK)	Type: Unsigned1: Valeurs nommées: 0: décalage positif par rapport à la fin de la DPDU émise/reçue; 1: décalage négatif par rapport à la fin de l'intervalle de temps.
CCAmode (indique de vérifier, oui ou non, la CCA avant émission)	Type: unsigned2 (voir 9.1.9.4.3): Valeurs nommées: 0: mode CCA 4; 1: mode CCA 1; 2: mode CCA 2; 3: mode CCA 3
KeepListening (indique s'il convient, oui ou non, que la DLE continue d'actionner périodiquement son récepteur jusqu'à la fin de l'intervalle de temps, même après qu'une DPDU ACK/NAK a été reçue; voir 9.1.9.4.7)	Type: Boolean1
Reserved (alignement d'octets)	Type: Unsigned2=0
XmitEarliest (temps le plus tôt pour commencer l'émission de DPDU; décalage par rapport au temps de début programmé de l'intervalle de temps)	Type: Unsigned16 Unités: 2^{-20} s
XmitLatest (temps le plus tard pour commencer l'émission de DPDU; décalage par rapport au temps de début programmé de l'intervalle de temps)	Type: Unsigned16 Unités: 2^{-20} s
WakeupAck (temps au plus tôt auquel la réception de DPDU ACK/NAK peut être censée commencer; active le récepteur suffisamment tôt pour recevoir une DPDU ACK/NAK commençant à cet instant. La sémantique dépend du champ ACKref: si AckRef=1, soustraire la valeur à la fin programmée de l'intervalle de temps pour déterminer WakeupAck)	Type: Unsigned16 Unités: 2^{-20} s
TimeoutAck (temps le plus tard auquel la réception d'une DPDU ACK/NAK peut être censée commencer. La DLE peut désactiver le récepteur si la réception de DPDU ACK/NAK n'a pas commencé au plus tard à cet instant. La sémantique dépend du champ ACKref: si AckRef=1, soustraire la valeur à la fin programmée de l'intervalle de temps pour déterminer TimeoutAck)	Type: Unsigned16 Unités: 2^{-20} s
NOTE Une valeur AckEarliest de 402 (384 μs) prend en charge l'exigence SIFS de l'IEEE 802.15.4 de 6 octets, plus 6 octets pour le SHR et le PHR d'une PhPDU avant le début de la PhSDU de la DPDU ACK/NAK.	

Le Tableau 164 spécifie la structure du modèle d'initiateur de transaction.

Tableau 164 – Structure du modèle d'initiateur de transaction

Nombre d'octets	bits							
	7	6	5	4	3	2	1	0
1	* Index (range 1..127)							
1	Type		AckRef	CheckCCAmode		KeepListening	Réservé	
2	XmitEarliest							
2	XmitLatest							
2	WakeupAck							
2	TimeoutAck							

Le Tableau 165, le Tableau 166 et le Tableau 167 spécifient les valeurs pour les trois modèles d'intervalle de temps de DLE par défaut. Ces modèles d'intervalle de temps en lecture seule utilisent les indices 1, 2, et 3, et doivent être utilisés pour des liaisons qui sont spécifiées dans des annonces. Leur structure est d'usage général et ils peuvent aussi être référencés par d'autres liaisons.

La DLE doit être capable d'émettre et de recevoir des ACK/NAK dans la temporisation de 1 032 µs ± 100 µs indiquée dans ces modèles par défaut, ou plus lentement si spécifiée dans un modèle d'intervalle de temps de remplacement. L'attribut dlmo.DeviceCapability.DAckTurnaround informe le gestionnaire de système si un appareil est capable de traiter plus rapidement des DPDU ACK/NAK.

Ces modèles par défaut sont exigés pour les processus initiaux de mise en service de l'appareil et de rattachement au sous-réseau. Il n'existe aucune exigence pour toute utilisation non rattachée dans un système opérationnel.

Tableau 165 – Modèle de répondeur de transaction par défaut, utilisé au cours du processus de rattachement

Champ	Valeur par défaut	Explication
* Index	1	-
Type	0	-
AckRef	0	-
RespectBoundary	1	-
WakeupDPDU	1 271	1 212 µs
TimeoutDPDU	3 578	3 412 µs
AckEarliest	977	932 µs
AckLatest	1 187	1 132 µs

Tableau 166 – Modèle d'initiateur de transaction par défaut, utilisé au cours du processus de rattachement

Champ	Valeur par défaut	Explication
* Index	2	-
Type	1	-
AckRef	0	-
CCAmode	1	-
KeepListening	0	-
XmitEarliest	2 319	2 212 µs
XmitLatest	2 529	2 412 µs
WakeupAck	977	932 µs
TimeoutAck	1 187	1 132 µs

Tableau 167 – Modèle de répondeur de transaction par défaut, utilisé au cours du processus de rattachement

Champ	Valeur par défaut	Explication
* Index	3	-
Type	0	-
AckRef	0	-
RespectBoundary	0	Si la réception de DPDU débute dans les limites de l'intervalle de temps, parachever le traitement de transaction
WakeupDPDU	0	Début d'intervalle de temps; permettant la prise en charge d'intervalles de temps qui sont contigus. Dans le premier intervalle de temps d'une série contiguë, un appareil peut insérer que le temps d'installation au début du premier intervalle de temps ne dépasse pas 1 271 µs
TimeoutDPDU	0xFFFF	Fin d'intervalle de temps
AckEarliest	977	Le même que pour le modèle de répondeur de transaction par défaut
AckLatest	1 187	Le même que pour le modèle de répondeur de transaction par défaut

9.4.3.3.3 Temporisations de modèle par défaut

Les modèles d'intervalle de temps par défaut visent à concorder avec la structure des intervalles de temps de l'IEC 62591.

Le temps d'émission de la DPDU Data par défaut est basé sur un décalage de 2 312 µs à partir du début de l'intervalle de temps jusqu'au début de la DPDU Data. Le modèle d'initiateur de transaction par défaut rend compte des ± 100 µs de la gigue d'initiateur de transaction, conduisant à une plage par défaut de 2 312 µs ± 100 µs. Le modèle de récepteur de transaction par défaut rend compte des ± 100 µs de gigue d'émission, plus une dérive d'horloge de ± 1 000 µs, conduisant à une plage de réception pour la DPDU Data de 2 312 µs ± 1 100 µs.

Le temps d'émission de la DPDU ACK/NAK par défaut est basé sur un décalage de 1 032 µs à partir de la fin de la DPDU Data jusqu'au début de la DPDU ACK/NAK. Les modèles de transaction d'intervalle par défaut permettent la prise en charge de ± 100 µs de gigue d'émission, pour un modèle par défaut de 1 032 µs ± 100 µs. La dérive d'horloge est considérée sans importance dans un court intervalle de temps à partir de la fin d'une DPDU Data jusqu'au début d'une DPDU ACK/NAK immédiatement consécutive.

Dans la radio selon l'IEEE 802.15.4 utilisée dans la présente norme, l'en-tête de couche physique a une durée de 6 octets, ou 192 µs. Ainsi, l'émission ou la réception radio

commence 192 μs plus tôt que les temps spécifiés dans les modèles. La CCA, si elle est exigée, précède le démarrage radio.

Les modèles ne rendent pas compte des éléments internes au fonctionnement radio, les laissant comme une question interne à un appareil. Par exemple, la valeur de 932 μs pour WakeupACK signifie que l'en-tête de couche physique peut commencer l'émission 192 μs plus tôt ou dès 932 μs -192 μs = 740 μs suivant la fin de la DPDU. Le récepteur de la DPDU ACK/NAK, aussi avec un WakeupAck nominal de 932 μs, a donc besoin que sa radio fonctionne à 740 μs suivant la fin de la DPDU et qu'elle soit prête alors à commencer à recevoir la DPDU ACK/NAK. Si le récepteur a besoin de plus de temps pour rendre compte du démarrage de la radio et de la gigue du récepteur, la radio a besoin d'être activée assez en avance de 740 μs pour assurer qu'elle peut recevoir l'en-tête complet de couche physique.

9.4.3.3.4 Considérations pour l'espace minimum exigé entre des émissions

Certains régimes de réglementation exigent qu'il y ait une période de temps minimale après qu'une émission a cessé avant que l'appareil ne puisse de nouveau émettre. Il est de la responsabilité de chaque DLE de noter le temps auquel l'émission la plus récente a cessé, d'utiliser cette information pour déterminer le moment au plus tôt auquel l'appareil peut émettre de nouveau et de s'abstenir d'activer son émetteur avant le moment en question.

NOTE Cela peut être réalisé en enregistrant séparément le temps de fin de l'émission la plus récente, en ajoutant une constante de délai Tx dépendant du mode de fonctionnement déclaré, puis en comparant ce temps qui en résulte avec le temps projeté de la prochaine émission pour déterminer si l'émission est autorisée dans le cadre des réglementations applicables. Voir l'Annexe V.

Alors que le gestionnaire de système peut configurer les divers dlmo.TsTemplates pour fournir un comportement en accord avec ces exigences de réglementation, il est de la responsabilité de l'appareil individuel d'assurer que les exigences sont respectées quelle que soit la configuration. L'attribut dlmo.CountryCode (9.1.15.6 et 9.4.2.19) fournit les informations de configuration nécessaires pour déterminer les aspects de réglementation en vigueur.

9.4.3.4 dlmo.Neighbor

9.4.3.4.1 Généralités

L'attribut dlmo.Neighbor est un ensemble OctetString indexé qui contient des informations relatives aux voisins immédiats en monodiffusion. Les entrées de dlmo.Neighbor sont référencées par des graphes et des liaisons.

Le gestionnaire de système insère, met à jour ou efface des entrées dlmo.Neighbor en envoyant au DMAP des entrées de voisins, avec un indice unique et (si sélectionné) un temps TAI de basculement. L'adresse 16 bits du voisin est utilisée comme indice.

Les voisins dans l'attribut de dlmo.Neighbor sont mis par le gestionnaire de système, pas par la DLE elle-même. La DLE construit de façon autonome une liste de voisins candidats dans l'attribut dlmo.Candidates, tel que décrit en 9.4.2.24. Cette liste est ensuite transmise au gestionnaire de système. Le gestionnaire de système considère la connectivité radio qui est rapportée dans dlmo.Candidates, mais peut également considérer d'autres critères tels que les contraintes de ressource, les performances historiques ou la topologie du sous-réseau D.

Lorsque la DLE traite une annonce au cours du processus de rattachement, la DLE ajoute automatiquement le routeur d'annonce comme entrée dans le tableau de dlmo.Neighbor, permettant ainsi la communication par l'intermédiaire du proxy. Cette entrée persiste après que la DLE a rejoint avec succès le sous-réseau D, à moins qu'elle ne soit supprimée ou mise à jour par le gestionnaire de système. Le gestionnaire de système infère l'existence et le contenu de cette entrée en se basant sur l'identité du proxy que la DLE utilise pour rejoindre le sous-réseau D. Voir 9.3.5.2.1.

La présente norme suit la convention de l'IETF RFC 4944, selon laquelle les adresses de 16-bits en monodiffusion sont limitées à la plage de 0x 0xxx xxxx xxxx xxxx. Voir 9.3.3.6.

L'information de diagnostic relative aux voisins peut être trouvée dans l'attribut dlmo.NeighborDiag (voir 9.4.3.9).

9.4.3.4.2 Sémantique

Le Tableau 168 spécifie les champs pour dlmo.Neighbor.

Tableau 168 – Champs de dlmo.Neighbor

Nom de champ	Codage de champ
* Index (DL16Address du voisin)	Type: ExtDLUint (utilisé comme un indice) Plage valide: 1..32 767
EUI64 (EUI64Address du voisin)	Type: EUI64Address
GroupCode1 (associer un code de groupe à un ensemble de voisins; utilisé par dlmo.Link)	Type: Unsigned8
GroupCode2 (associer un code de groupe à un ensemble de voisins; utilisé par dlmo.Link)	Type: Unsigned8
ClockSource (indique si le voisin est une source d'horloge de DL)	Type: Unsigned2 Valeurs nommées: 0: pas une source d'horloge 1: secondaire 2: préférentielle 3: réservée
ExtGrCnt (nombre total de graphes virtuellement étendus pour ce voisin)	Type: Unsigned2
DiagLevel (sélection de diagnostics de voisins à recueillir)	Type: BooleanArray2 Indices nommés: 0: recueillir des diagnostics de liaison 1: recueillir des diagnostics d'horloge
LinkBacklog (indique que les informations relatives à la liaison sont fournies pour faciliter l'effacement de l'accumulation dans la file d'attente de messages pour le voisin)	Type: Unsigned1; Valeurs nommées: 0: aucune information supplémentaire relative à la liaison 1: des informations supplémentaires relatives à la liaison sont fournies
Reserved (alignement d'octets)	Type: Unsigned1=0
ExtendGraph	Type: SEQUENCE OF Octet2 (SIZE ExtGrCnt) – paires d'octets Plage valide: Voir Tableau 170
LinkBacklogIndex (activer cette liaison pour effacer l'accumulation de file d'attente)	Type: ExtDLUint Plage valide: 1..127 Null (vide) et n'est pas émis si LinkBacklog=0 Indice de liaison si LinkBacklog=1
LinkBacklogDur (durée de l'activation de liaison)	Type: Unsigned8 Unités: Occurrences de liaison Null (vide) et n'est pas émis si LinkBacklog=0 1..255 si LinkBacklog=1
LinkBacklogActivate (critère d'activation de liaison)	Type: Unsigned8 Unités: DPDU en file d'attente Null (vide) et n'est pas émis si LinkBacklog=0 1..255 si LinkBacklog=1

Le Tableau 169 montre la structure de dlmo.Neighbor.

Tableau 169 – Structure de dlmo.Neighbor

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1–2	* Index							
8	EUI64							
1	GroupCode1							
1	GroupCode2							
1	ClockSource.		ExtGrCnt		DiagLevel		LinkBacklog	Reserved
2 × ExtGrCnt	$ExtendGraph_1$							
	...							
	$ExtendGraph_{ExtGrCnt}$							
0..1	LinkBacklogIndex							
0..1	LinkBacklogDur							
0..1	LinkBacklogActivate							

Les champs comprennent:

- EUI64. Afin de communiquer avec un voisin, le DSC a besoin de l'adresse EUI64 du voisin en question. Cette information est stockée dans la table Neighor. Elle est peuplée par le gestionnaire de système, à une seule exception près. Pendant les processus de rattachement et de mise en service, où l'entrée de voisin est créée automatiquement à partir de l'annonce, EUI64Address doit être acquise par la DLE par l'intermédiaire de la DPDU ACK/NAK selon la description en 9.1.10.1.
- GroupCode1, GroupCode2. Les liaisons avec un code de groupe concordant peuvent être utilisées pour s'adresser à ce voisin. La portée du code de groupe se situe au sein d'une seule DLE. Lorsqu'une liaison a un groupe NeighborType=2, la liaison indique un code de groupe au lieu d'un voisin, et la liaison s'applique à toute DPDU placée en file d'attente où le voisin a un code de groupe concordant. Cela permet à une seule et même liaison d'émission d'être partagée par un groupe de voisins. Une valeur de zéro indique qu'aucun code de groupe ne s'applique. La prise en charge des codes de groupe est obligatoire dans les routeurs, mais il s'agit d'une option de construction dans des appareils E/S. La présence ou l'absence de cette capacité est rapportée au gestionnaire de système lorsque la DLE rejoint le sous-réseau D par l'intermédiaire du champ dlmo.DeviceCapability.ConstructionOptions (voir 9.4.2.23).
- ClockSource. Si cet indicateur est > 0, le voisin doit être une source d'horloge de DL pour cette DLE. Une valeur de 1 indique une source d'horloge de DL secondaire, et une valeur de 2 indique une source d'horloge de DL préférentielle. Voir 9.1.9.2.3.
- ExtGrCnt et ExtendGraph. Voir 9.1.6.3 pour un débat relatif aux extensions de graphe. Voir Tableau 170 pour les champs dans chaque entrée d'extension de graphe dans ExtendGraph. Si l'adresse du voisin concorde avec l'adresse de destination codée dans le sous-en-tête DADDR, et le Graph_ID indiqué dans ExtendGraph concorde au chemin de DL de la DPDU, prolonger le graphe désigné jusqu'au voisin en question pour la DPDU en question. Pour chaque voisin, jusqu'à trois graphes de ce type peuvent être prolongés. La prise en charge d'ExtendGraph est une option de construction d'appareil, et la présence ou l'absence de cette capacité est rapportée au gestionnaire de système lorsque la DLE rejoint le sous-réseau D par l'intermédiaire du champ dlmo.DeviceCapability.ConstructionOptions (voir 9.4.2.23).

Pour chaque entrée de ExtendGraph, inclure:

– GraphID est l'identificateur de graphe de 12 bits qui est prolongé. Voir 9.4.3.6.

– LastHop. Si cet indicateur est 1, la DLE doit seulement utiliser des liaisons vers le voisin pour des DPDU applicables. Dans ce cas, la DLE doit traiter l'indice de graphe prolongé comme la seule alternative de transmission.

– PreferredBranch. Si cet indicateur est 1, il convient que la DLE traite cette extension de graphe comme la branche préférentielle pour les DPDU applicables. (Voir le champ PreferredBranch en 9.4.3.6.2).

Le Tableau 170 spécifie les champs pour chaque élément dans la séquence ExtendGraph.

Tableau 170 – Champs de ExtendGraph

Nom de champ	Codage de champ
Graph_ID (*Indice de l'attribut dlmo.Graph)	Type: Unsigned12
LastHop (indique si le voisin doit être le dernier saut)	Type: Boolean1
PreferredBranch (indique s'il faut traiter le voisin comme étant la branche préférentielle)	Type: Boolean1
Reserved (alignement d'octets)	Type: Unsigned2=0

Les graphes sont prolongés implicitement, indépendamment du fait que ExtendGraph soit indiqué ou pas. La caractéristique ExtendGraph explicite facultative vise à prendre en charge les optimisations qui cherchent à commander l'ID de graphe de ce saut final, et/ou le désigner comme étant le dernier saut ou la branche préférentielle.

– DiagLevel. Si cet indicateur a un nombre quelconque de bits différents de zéro, la DLE doit recueillir les diagnostics de liaison pour ce voisin dans l'attribut en lecture seule dlmo.NeighborDiag. Si Bit0=1, les diagnostics de résumés doivent être accumulés. Si Bit1=1, les diagnostics d'horloge doivent être accumulés. Voir 9.4.3.9;
– LinkBacklog, LinkBacklogIndex, LinkBacklogDur, et LinkBacklogActivate fournissent un indice à une liaison qui peut être activée par l'intermédiaire du sous-en-tête DAUX (type 2). Voir 9.3.5.4 pour une description générale relative à l'activation de liaison. La DLE, lorsqu'elle émet une DPDU vers un voisin, peut détecter une accumulation de DPDU applicables dans sa file d'attente de messages, et donc signaler au voisin d'activer la liaison Index=LinkBacklogIndex pour recevoir des DPDU pour les prochaines occurrences de LinkBacklogDur de la liaison (voir 9.3.5.4). LinkBacklogActivate indique la taille de l'accumulation de DPDU applicables dans la file d'attente, à l'exclusion de la DPDU en cours d'émission, qu'il convient de déclencher via l'envoi du DAUX d'activation de liaison, à moins que la liaison ne soit déjà activée.

Il convient que le gestionnaire de système, lorsqu'il configure LinkBacklogIndex, LinkBacklogDur et LinkBacklogActivate, configure également une liaison d'émission avec le même indice; de sorte qu'une liaison de réception sur le voisin a le même indice qu'une liaison d'émission correspondante dans la DLE qui est à l'origine du message d'activation de liaison, les deux étant activées pour le nombre d'occurrences indiquées par LinkBacklogDur. Il convient que les DPD placées en file d'attente à destination de ce voisin reçoivent une haute priorité pour cet indice de liaison au cours de la période définie par LinkBacklogDur. Lorsqu'elle compte les DPDU candidates dans la file d'attente de messages, il convient que la DLE rende compte de toutes les DPDU placées en file d'attente qui peuvent être adressées au voisin.

Il convient que l'initiateur de transaction (la DLE qui initie l'activation des liaisons inactives) active sa propre liaison inactive pour un nombre total d'occasions d'émission de LinkBacklogDur (occurrences de liaison). Il convient que le récepteur (la DLE sur laquelle la liaison de réception inactive a été activée) active sa liaison inactive pour des occasions de réception de LinkBacklogDur (occurrences de liaison). Il convient que les occasions d'émission et de réception soient comptées même si une liaison de plus haute priorité est réellement utilisée dans un intervalle de temps particulier. En général, il convient que la liaison inactive soit configurée avec une haute priorité.

Le Tableau 171 spécifie la structure de ExtGraph.

Tableau 171 – Structure de ExtGraph

Nombre d'octets	bits							
	7	6	5	4	3	2	1	0
1	Graph_ID (bits 11..4)							
1	Graph_ID (bits 3..0)				LastHop	PreferredBranch	Reserved = 0	

9.4.3.4.3 dlmo.NeighborDiagReset

L'attribut dlmo.NeighborDiagReset fournit l'accès en lecture/écriture au champ de l'attribut de dlmo.Neighbor qui est utilisé pour établir le niveau de diagnostic de voisin. Il est conceptuellement similaire à une vue SQL de dlmo.Neighbor. Il peut être utilisé pour lire et écrire un champ spécifique au sein de dlmo.Neighbor, mais il ne doit pas être utilisé pour ajouter ou supprimer des entrées.

Le Tableau 172 spécifie les champs au sein de l'OctetString dlmo.NeighborDiagReset.

Tableau 172 – Champs de dlmo.NeighborDiagReset

Nom de champ	Codage de champ
* Index	Type: ExtDLUint (utilisé comme un indice) Plage valide: 1..32 767
Reserved (alignement d'octets)	Unsigned4 = 0
DiagLevel (sélection de diagnostics de voisins à recueillir)	Type: BooleanArray3; Indices nommés: 0: recueillir des informations de résumé 1: recueillir des informations d'horloge
Reserved (alignement d'octets)	Type: Unsigned2 = 0

Le Tableau 173 montre la structure de dlmo.NeighborDiagReset.

Tableau 173– Structure de dlmo.NeighborDiagReset

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1..2	* Index							
1	Reserved = 0				DiagLevel		Reserved = 0	

Les champs sont exactement tels que spécifiés pour les champs de même nom dans l'attribut dlmo.Neighbor, et les valeurs instantanées de ces champs sont les mêmes que dans dlmo.Neighbor.

9.4.3.5 dlmo.Superframe

9.4.3.5.1 Généralités

dlmo.Superframe est un ensemble OctetString indexé qui contient des supertrames. La structure de supertrames permet au gestionnaire de système de relier un jeu de liaisons (dlmo.Link) à un programme répétitif d'intervalle de temps (dlmo.Superframe) de durée fixe. La supertrame désigne également un programme de saut de voie de ligne de base pour ces liaisons.

Le gestionnaire de système insère, met à jour ou efface des entrées de dlmo.Superframe en envoyant au DMAP une supertrame, accompagnée d'un indice unique et, si sélectionné, d'un temps TAI de basculement.

Chaque supertrame décrit un programme qui est spécifié en référence au temps TAI zéro. La dérivation de l'état d'intervalle de temps courant à partir de la définition d'une supertrame est décrite en 9.4.3.5.3.

Une supertrame peut être configurée avec des temporisations randomisées, destinées exclusivement aux liaisons qui émettent ou reçoivent des sollicitations et/ou des annonces. Sachant qu'un programme de supertrames randomisées n'est pas synchronisé par rapport à ses voisins, une telle supertrame ne doit pas être utilisée pour émettre une charge utile dans des DSDU.

9.4.3.5.2 Sémantique

Le Tableau 174 spécifie les champs pour dlmo.Superframe.

Tableau 174 – Champs de dlmo.Superframe

Nom de champ	Codage de champ
* Index	Type: ExtDLUint (utilisé comme un indice) Plage valide: 1..127
TsDur (durée des intervalles de temps au sein de la supertrame; les intervalles de temps sont réalignés avec une référence de temps TAI toutes les 250 ms)	Type: Unsigned16 Unités: 2^{-20} s
ChIndex (sélectionne le modèle de saut de voie dans dlmo.Ch)	Type: ExtDLUint
ChBirth (numéro d'intervalle de temps absolu auquel le modèle de saut de voie a nominalement commencé)	Type: Unsigned8
SfType (type de supertrame)	Type: Unsigned2; Valeurs nommées: 0: ligne de base 1: saut sur liaison seulement 2: randomiser la durée de saut lent 3: randomiser la période de supertrame
Priority (priorité pour sélectionner parmi plusieurs liaisons disponibles)	Type: Unsigned4
ChMapOv (indique s'il faut neutraliser la valeur par défaut de ChMap)	Type: Boolean1; Plage valide: false: ChMapOv non émis, donc prenant la valeur par défaut égale à 0x7FFF; true: ChMapOv émis et utilisé
IdleUsed	Type: Boolean1; Plage valide: false: IdleTimer non émis, donc prenant la valeur par défaut égale à -1; true: IdleTimer émis et utilisé
SfPeriod (nombre de base d'intervalles de temps dans chaque cycle de supertrame)	Type: Unsigned16 Plage valide: > 0
SfBirth (numéro d'intervalle de temps absolu auquel le premier cycle de supertrame a nominalement commencé)	Type: Unsigned16
ChRate (indique le nombre d'intervalles de temps par saut)	Type: ExtDLUint Plage valide: 0 = invalide 1 = saut de voie discrétisé > 1 = saut de voie lent [a]
ChMap (carte de voies utilisée pour éliminer certaines voies du modèle de saut de voie afin de limiter le spectre de fréquences utilisé)	Type: Unsigned16 ou null
IdleTimer (temporisateur inactivité/éveil pour supertrame)	Type: Integer32 ou null Voir texte
RndSlots (indique l'étendue de la randomisation, en nombre d'intervalles de temps)	Type: Unsigned8 ou null
a Dans certains régimes de réglementation, tels que spécifiés par l'intermédiaire de dlmo.CountryCode (9.4.2.19), la valeur maximale de ChRate est contrainte à être inférieure ou égale à (400 ms / TsDur).	

Le Tableau 175 montre la structure de dlmo.Superframe.

Tableau 175 – Structure de dlmo.Superframe

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	* Index							
2	TsDur							
1	ChIndex (plage 1..127)							
1	ChBirth							
1	SfType		Priority				ChMapOv	IdleUsed
2	SfPeriod							
2	SfBirth							
1..2	ChRate							
0 ou 2	ChMap							
0 ou 4	IdleTimer							
0 ou 1	RndSlots							

Les champs comprennent:

- durée d'intervalle de temps (dlmo.Superframe[ ].TsDur). Tous les intervalles de temps dans une supertrame ont la même durée, et une DLE ne doit pas traiter plusieurs durées d'intervalles de temps en même temps. Les intervalles de temps doivent être réalignés à l'horloge TAI toutes les 250 ms. Voir 9.1.9.1 pour des informations relatives à l'alignement d'intervalles de temps;
- identificateur de modèle de saut de voie (dlmo.Superframe[ ].ChIndex). Sélectionner un modèle disponible dans dlmo.Ch (voir 9.4.3.2.2);
- anniversaire de saut de voie (dlmo.Superframe[ ].ChBirth). Spécifie le point de départ du modèle de saut de voie, sous la forme d'un décalage d'intervalle de temps par rapport à TAI=0. Le calcul de la position courante dans la séquence de saut est décrit en 9.4.3.5.3;
- le type de supertrame (dlmo.Superframe[ ].SfType) indique le type de supertrame. Le traitement de chaque type de supertrame est décrit en 9.4.3.5.3;
- la priorité de supertrame (dlmo.Superframe[ ].Priority) indique la priorité de la supertrame. Une plus grande valeur de Priority donnera à ces liaisons de supertrame une plus haute priorité. Voir 9.1.8.5 (pseudocode) pour des informations complémentaires relatives aux niveaux de priorité;
- valeur par défaut de neutralisation de carte de voies (dlmo.Superframe[ ].ChMapOv). Indique de neutraliser, oui ou non, la valeur par défaut de ChMap de 0x7FFF. Si ChMapOv=1, changer la valeur par défaut basée sur ChMap;
- la période de supertrame (dlmo.Superframe[ ].SfPeriod) indique le nombre des intervalles de temps dans chaque cycle de base de la supertrame. Avec des intervalles de temps de 10 ms, une période de supertrame de 16-bits prend en charge des supertrames d'une longueur pouvant atteindre 10 minutes;
- l'anniversaire de supertrame (dlmo.Superframe[ ].SfBirth) indique le point de départ nominal pour la "première" supertrame qui a commencé son premier cycle peu après TAI=0. Il fournit le décalage d'intervalle de temps par rapport à l'intervalle de temps absolu 0, qui s'est produit au temps nominal TAI=0. SfBirth doit être égal à ChBirth lorsque fType= 1. Voir 9.4.3.5.3;
- la fréquence de saut de voie (dlmo.Superframe[ ].ChRate) indique le nombre des intervalles de temps par saut. Une fréquence de saut de voie 1 indique un saut de voie discrétisé; une fréquence de saut de voie supérieure à 1 indique un certain degré de saut de voie lent. ChRate doit être = 1 lorsque SfType= 1;
- la carte de voies (dlmo.Superframe[ ].ChMap) est utilisée pour éliminer certaines voies du modèle de saut de voie, ce qui raccourcit le modèle de saut de voie en utilisation.

Les positions de bits 0..15 correspondent aux voies 0..15, où un bit de 0 en n'importe quelle position indique que la voie correspondante ne doit pas être utilisée par la supertrame, avec la séquence de saut de voie raccourcie en conséquence. Cet attribut ne doit pas être émis et prend par défaut la valeur 0x7FFF si ChMapOv=0 (à savoir: la voie facultative 15 est exclue par défaut);

- le temporisateur de supertrame inactive (dlmo.Superframe[ ].IdleUsed, dlmo.Superframe[ ].IdleTimer) fournit au gestionnaire de système le contrôle sur l'instant où une supertrame est activée ou inactive, où une supertrame inactive traite toutes ses liaisons comme étant inactives. Le gestionnaire de système peut positionner IdleTimer par l'intermédiaire de l'attribut dlmo.Superframe ou de l'attribut dlmo.SuperframeIdle. IdleTimer n'est pas émis et prend par défaut une valeur de -1 si IdleUsed=0;
- lorsque IdleTimer est mis à un nombre positif, la supertrame doit être active et IdleTimer doit être décrémenté par la DLE toutes les secondes TAI jusqu'à ce qu'il atteigne une valeur de 0. Lorsque la valeur de IdleTimer est 0, la supertrame doit être inactive;
- lorsque IdleTimer est mis à un nombre négatif qui est inférieur à -1, la supertrame doit être inactive et IdleTimer doit être incrémenté par la DLE toutes les secondes TAI jusqu'à ce qu'il atteigne une valeur de -1. Lorsque la valeur de IdleTimer est -1, la supertrame doit être active.

La randomisation des supertrames est commandée par dlmo.Superframe[ ].RndSlots. RndSlots est sans signification et ne doit pas être émise si dlmo.Superframe[ ].SfType<2. Les supertrames randomisées visent exclusivement à permettre des processus randomisés de découverte de sous-réseaux D. Par exemple, une DLE dans l'état de mise en service peut être configurée avec une supertrame randomisée qui est utilisée pour rechercher des annonces issues d'un sous-réseau D cible. Une telle randomisation peut être utilisée pour garantir que le cycle de sommeil du balayage n'est pas synchronisé avec le programme d'annonce du sous-réseau D cible, assurant ainsi qu'une annonce est finalement reçue. Il convient que seules les liaisons de réception, les annonces dédiées et les sollicitations soient configurées pour être utilisées avec une supertrame où RndSlots>0. Toutes les autres liaisons pour de telles supertrames randomisées doivent être traitées comme étant inactives.

– lorsque SfType=2, un nombre randomisé d'intervalles de temps dans la plage de 0 à RndSlots doit être ajouté à la fin de chaque cycle de supertrame;
– lorsque SfType=3, un nombre randomisé d'intervalles de temps dans la plage de 0 à RndSlots doit être ajouté à la longueur de chaque période de saut de voie lent. Si le saut de voie discrétisé est utilisé, chaque saut doit être traité comme une période de saut de voie lent prolongée par un nombre randomisé d'intervalles de temps.

9.4.3.5.3 Etat courant des intervalles de temps de supertrame

L'état courant des intervalles de temps de supertrame doit être dérivé des champs de supertrame tels que décrits dans le présent document. Cette description est censée ne pas contraindre les mises en œuvre, mais seulement les résultats. Toutes les caractéristiques décrites dans le présent document doivent être prises en charge par toutes les DLE qui sont conformes à la présente norme, à moins d'être spécifiquement indiquées comme étant une option de construction.

Ces dérivations de l'état courant d'intervalle de temps utilisent des champs dans dlmo.Superframe [ ], qui sont basés sur l'état au temps TAI zéro ou peu après. Les mises en œuvre peuvent raisonnablement utiliser ces formules pour établir un état de début lorsque la supertrame est initialisée, et ensuite mettre à jour cet état en allant vers l'avant par incréments. Cependant, l'approche de la mise à jour par incréments ne fonctionnera pas lorsqu'il y a un changement dans un champ quelconque utilisé dans le calcul de base, ou dans des champs dans d'autres attributs (dlmo.Ch [ ] en général et dlmo.Link [ ] pour SfType=1) qui sont utilisés dans le calcul de base. Lorsque ces champs sont modifiés, l'état courant a besoin d'être dérivé à nouveau.

NOTE La notion d'un intervalle de temps absolu est utilisée ici comme une variable pour calculer l'état courant d'intervalle de temps. D'autres normes utilisent un intervalle de temps absolu pour identifier l'intervalle de temps. La présente norme utilise un intervalle de temps absolu seulement comme une valeur intermédiaire dans un calcul; il n'est référencé nulle part ailleurs dans la présente norme.

Chaque période de quart de seconde TAI a un nombre fixe d'intervalles de temps qui peuvent être décrits par la formule:

SlotsPer0_25s = floor((2^{18} s) / TsDur)

où floor(x) est le plus grand nombre entier qui n'est pas supérieur à x.

Un nombre d'intervalle de temps absolu (SlotNumAbs) peut être dérivé du temps de départ programmé du courant intervalle de temps (ScheduledTaiTime), simplifié par le fait que ScheduledTaiTime doit être réaligné au temps TAI tous les quarts de seconde. Le décalage d'intervalle de temps par rapport à TAI=0 peut être dérivé en conséquence:

Tai0_25sStart = floor(ScheduledTaiTime / (0,25 s)) × (0,25 s)

SlotWithin0_25s = (ScheduledTaiTime – Tai0_25sStart) / TsDur

SlotNumAbs = ((Tai0_25sStart / (0,25 s)) × SlotsPer0_25s) + SlotWithin0_25s

SfType=0 désigne le cas de ligne de base, où tous les cycles de supertrame incluent un nombre fixe d'intervalles de temps et le programme de saut de voie a également un cycle fixe.

La superframe fournit une période de supertrame fixe (SfPeriod) qui est le nombre d'intervalles de temps dans chaque cycle de supertrame. Elle fournit également un nombre d'intervalles de temps absolu de (SfBirth), suivant TAI=0, comme temps de départ de référence pour la première supertrame. Le décalage de supertrame du courant intervalle de temps est:

SfOffset = (SlotNumAbs – SfBirth) mod SfPeriod

où (x mod y) est égal à (x – (floor(x / y) × y)) pour y positif.

Le modèle de saut de voie commence nominalement au nombre d'intervalles de temps absolu ChBirth. Le nombre d'éléments dans le programme de saut de voie (ChCount) peut être déterminé à partir de la taille du modèle de saut de voie sélectionné par ChIndex, et en soustrayant le nombre d'entrées qui sont enlevées de la séquence tel qu'indiqué par ChMap.

Le nombre d'intervalles de temps dans un cycle du modèle de saut de voie dépend de la question de savoir si le saut de voie lent ou le saut de voie discrétisé est utilisé, tel qu'indiqué par ChRate.

ChCycle = ChCount × ChRate

Le décalage d'intervalle de temps dans ce saut de voie est:

ChOffset = (SlotNumAbs – ChBirth) mod ChCycle

SfType=1 désigne une variante de saut de voie discrétisé, dans laquelle le saut de voie ne se produit que lorsqu'il y a une liaison.

La prise en charge d'appareil pour SfType=1 est une option de construction, telle que rapportée au gestionnaire de système par l'intermédiaire de l'attribut dlmo.DeviceCapability.ConstructionOptions (bit 5). SfType=1 ne doit pas être combiné avec le saut de voie lent, à savoir ChRate=1.

Le décalage de supertrame est tel que décrit dans SfType=0.

Le nombre de sauts de voie par cycle de supertrame (ChPerSuperframe) est déterminé en comptant le nombre des intervalles de temps dans chaque cycle de supertrame qui sont

référencés par au moins une liaison. Il ne s'agit pas d'un simple compte de liaisons, parce que certaines liaisons peuvent se référer à plusieurs intervalles de temps, et certains intervalles de temps peuvent être référencés par plusieurs liaisons.

Sachant que le nombre de sauts de voie est un multiple du nombre de supertrames, la prochaine étape est de calculer le nombre de cycles de supertrame qui ont été accomplis depuis TAI=0.

CurrentSfStartAbs = (SlotNumAbs – SfOffset)

SfCyclesSinceBirth = (CurrentSfStartAbs – SfBirth) / SfPeriod

Pour SfType=1, les cycles de cycle de supertrame et de saut de voie doivent débuter en même temps (SfBirth=ChBirth). Le décalage de voie au début de la supertrame courante est une fonction du nombre de voies mises en correspondance (voir 9.4.2.12). Il est déterminé par la formule:

$$\text{ChOffset}_{\text{StartSf}} = (\text{SfCyclesSinceBirth} \times \text{ChPerSuperframe}) \bmod \text{NumMappedChannels}$$

où NumMappedChannels = $\sum \text{dlmo.Superframe}[\,].\text{ChMap}_{k}$ pour la supertrame spécifique.

En partant de $\text{ChOffset}_{\text{StartSf}}$, le décalage de saut de voie courant peut être déterminé en parcourant la supertrame du début du cycle de supertrame jusqu'à l'intervalle de temps courant. Le décalage de saut de voie au sein de chaque cycle de supertrame ne peut pas être réduit à une simple formule linéaire simple, car les liaisons ne sont pas nécessairement uniformément étalées sur le cycle.

SfType=2 prolonge chaque intervalle de saut de voie lent par un nombre randomisé d'intervalles de temps dans la plage de 0 à dlmo.Superframe[ ].RndSlots. Le point de départ initial de la séquence de saut de voie peut également être randomisé lorsque SfType=2, et la temporisation de supertrame doit être telle que décrite pour SfType=0.

SfType=3 prolonge chaque cycle de supertrame par un nombre randomisé d'intervalles de temps dans la plage de 0 à dlmo.Superframe[ ].RndSlots. Il convient que le point de départ initial du cycle de supertrame soit également randomisé lorsque SfType=3, et la séquence de saut de voie doit être telle que décrite pour SfType=0.

9.4.3.5.4 Saut de voie lent

Le saut de voie lent est défini comme étant une supertrame où dlmo.Superframe[ ].ChRate>1, ce qui conduit à un ensemble de liaisons contiguës sur la même voie. La fréquence de saut de voie, ChRate, peut être configurée comme étant égale à la période de supertrame, SfPeriod, auquel cas chaque période de saut de voie peut raisonnablement être configurée comme étant une plage de liaisons utilisant dlmo.Link[ ].Schedule=2.

Il convient que le côté réception d'une configuration de saut de voie lent utilise le modèle de récepteur de transaction par défaut pour le balayage conformément au Tableau 167, ou un modèle similaire. Il convient que ce modèle, lorsqu'il est appliqué à des intervalles de temps contigus sur la même voie, actionne son récepteur sans interruption, et il peut exécuter une transaction jusqu'à son parachèvement même si cette transaction traverse la frontière d'un intervalle de temps. Une liaison de réception utilisant ce modèle peut se répéter fréquemment ou sans interruption dans une supertrame, habituellement avec une priorité basse pour donner la préséance aux opérations de saut de voie discrétisé.

Un ensemble de liaisons de réception de saut de voie lent sur une voie donnée, utilisant le modèle de récepteur de transaction par défaut pour le balayage, peut être temporairement interrompu par des transactions de plus haute priorité, par exemple, comme représenté graphiquement à la Figure 66. En l'absence de telles transactions, il convient que le récepteur fonctionne sans interruption à travers les frontières d'intervalle de temps dans de telles configurations.

Il convient que le côté émission d'une configuration de saut de voie lent utilise un modèle approprié pour une transaction d'émission sur le sous-réseau D, tel que le modèle d'initiateur de transaction par défaut conformément au Tableau 166. Il convient que la configuration de liaison d'émission soit configurée pour rendre compte de la dérive d'horloge. Par exemple, dans un sous-réseau D avec des intervalles de temps de 10 ms, un appareil particulier dans une configuration de saut de voie lent pourrait être censé subir une dérive d'horloge pouvant atteindre 15 ms entre les mises à jour d'horloge. Dans cet exemple, il convient de ne pas indiquer les deux premiers et deux derniers intervalles de temps dans chaque période de saut de voie lent comme étant des liaisons d'émission, incorporant ainsi des temps de garde de 20 ms dans la configuration.

Le côté émission dans une configuration de saut de voie lent peut désigner des intervalles de temps spécifiques pour l'émission, ou en variante, il peut désigner une plage d'intervalles de temps. Lorsqu'une plage d'intervalle de temps est indiquée, il convient que la fréquence de saut de voie concorde avec la période de supertrame, et il convient que la liaison d'émission soit désignée comme étant une plage (dlmo.Link[].Schedule=2), représentée graphiquement à la Figure 72. Dans cette configuration, il convient que la DLE traite chaque plage comme une seule occasion d'émission et sélectionne la liaison d'émission au sein de la plage sur une base randomisée.

9.4.3.5.5 dlmo.SuperframeIdle

dlmo.SuperframeIdle fournit l'accès en lecture/écriture aux seuls champs de dlmo.Superframe qui sont relatifs à la supertrame inactive. Il est conceptuellement similaire à une vue SQL de dlmo.Superframe. Il peut être utilisé pour lire et écrire des champs spécifiques dans des OctetStrings indexés de supertrame, mais il ne peut pas être utilisé pour ajouter ou supprimer des entrées.

Le Tableau 176 spécifie les champs au sein de l'OctetString dlmo.SuperframeIdle.

Tableau 176 – Champs de dlmo.SuperframeIdle

Nom de champ	Codage de champ
* Index	Type: ExtDLUint (utilisé comme un indice)
Reserved (alignement d'octets)	Type: Unsigned7=0
IdleUsed (indique si la supertrame est inactive lorsque la valeur de IdleTimer est zéro)	Type: Boolean1
IdleTimer (temporisateur inactivité/éveil pour supertrame)	Type: Integer32 ou null

Le Tableau 177 montre la structure de dlmo.SuperframeIdle.

Tableau 177 – Structure de dlmo.SuperframeIdle

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	* Index							
1	Reserved = 0							IdleUsed
0 ou 4	IdleTimer							

Les champs sont exactement tels que spécifiés pour les champs de même nom dans l'attribut dlmo.Superframe, et les valeurs instantanées de ces champs sont identiques à celles contenues dans dlmo.Superframe.

9.4.3.6 dlmo.Graph

9.4.3.6.1 Généralités

dlmo.Graph est un ensemble OctetString indexé qui contient des graphes. Sur une DLE particulière, un graphe est simplement une liste de voisins qui peuvent être utilisés pour le prochain saut lorsqu'un graphe est spécifié dans le sous-en-tête DROUT.

Le gestionnaire de système insère, met à jour ou efface des entrées de dlmo.Graph en envoyant au DMAP un graphe, accompagné d'un indice unique et (si sélectionné) d'un temps TAI de basculement.

Graph ID = 0 ne doit pas être utilisé, car cette valeur est réservée comme un indicateur dans le sous-en-tête DROUT, tel que décrit en 9.3.3.6.

Les ID de graphe sont limités à une plage de valeurs de 12 bits, avec une plage de $1..2^{12}$. Dans le routage de source, ces ID de graphe de 12 bits sont codés sous la forme 0x 1010 gggg gggg gggg.

Comme décrit en 9.1.6.3, des voisins immédiats sont implicitement traités comme étant couverts par un graphe, indépendamment du fait que le voisin soit énuméré dans la structure de graphe ou non. Lorsque l'adresse de destination de la DPDU est dans une table de voisins d'une DLE, le graphe est automatiquement étendu pour couvrir le voisin en question. Ainsi, même si la structure de dlmo.Graph ne peut prendre en charge qu'un petit nombre de voisins, le graphe peut en traiter beaucoup plus par l'intermédiaire de ces extensions de graphe.

Il convient que les ID de graphe soient uniques au sein de la portée d'un sous-réseau D. Cependant, des ID de graphe en doublons ne sont pas interdits. Lorsque deux graphes avec le même ID de graphe se rencontrent dans une seule et même DLE, ils s'unissent pour devenir un seul graphe.

9.4.3.6.2 Sémantique

Le Tableau 178 spécifie les champs pour dlmo.Graph.

Tableau 178 – dlmo.Graph

Nom de champ	Codage de champ
* Index	Type: ExtDLUint (utilisé comme un indice) Plage valide: 1..4095
PreferredBranch (indique s'il faut traiter le premier voisin énuméré comme étant la branche préférentielle)	Type: Boolean1
NeighborCount	Type: Unsigned3 Plage valide: 0..4
Queue (alloue des tampons dans la file d'attente de messages pour les DPDU qui sont transmises à l'aide de ce graphe)	Type: Unsigned4
MaxLifetime	Type: ExtDLUint
Neighbors (indice dans dlmo.Neighbor; habituellement deux ou trois voisins dans la liste pour la diversité de prochain saut)	Type: SEQUENCE OF ExtDLUint (SIZE(NeighborCount))

Le Tableau 179 montre la structure de dlmo.Graph dans le cas où chaque ExtDLUint exigerait un octet.

Tableau 179 – Structure de dlmo.Graph

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1..2	* Index							
1	PreferredBranch	NeighborCount			Queue			
1..2	MaxLifetime							
1..2	$Neighbors_0$							
1..2	$Neighbors_1$							
...	...							
1..2	NeighborsNeighborCount-1							

Les éléments comprennent:

- dlmo.Graph[ ].PreferredBranch. Si cet indicateur est 1, traiter le premier voisin énuméré comme étant la branche préférentielle, et il convient que la DLE attende jusqu'à ce qu'il y ait une occasion pour essayer au moins une émission le long de la branche préférentielle avant de tenter d'autres solutions de remplacement. Si cet indicateur est 0, ne pas accorder un tel traitement préférentiel au premier voisin énuméré.
- dlmo.Queue permet au gestionnaire de système de réserver jusqu'à 15 tampons de la file d'attente de messages pour les DPDU qui suivent le graphe.
- dlmo.Graph[ ].MaxLifetime (unités ¼ s). Si cet élément est différent de zéro, la valeur de dlmo.MaxLifetime doit être neutralisée pour toutes les DPDU transmises en suivant ce graphe.
- Liste de voisins (communément, deux voisins pour la diversité de liaison de prochain saut).

9.4.3.7 dlmo.Link

9.4.3.7.1 Généralités

dlmo.Link est un ensemble OctetString indexé qui contient des liaisons. Chaque liaison se réfère à exactement une entrée de dlmo.Superframe.

Le gestionnaire de système insère, met à jour ou efface des liaisons en envoyant au DMAP une liaison, avec un indice unique et (si sélectionné) d'un temps TAI de basculement.

Quand un voisin est référencé dans une liaison d'émission, les DPDU qui se réfèrent à ce voisin sont considérées comme des candidats pour la liaison. Les DPDU qui se réfèrent au voisin via la première entrée dans le sous-en-tête DROUT, directement par l'adresse ou indirectement par l'intermédiaire d'un graphe, doivent être considérées comme étant des candidats pour la liaison. En outre, les DPDU qui désignent le voisin comme l'adresse de destination dans le sous-en-tête DADDR doivent également être considérées comme étant des candidats pour la liaison. L'exception est que certaines liaisons sont désignées exclusivement pour les DPDU suivant des graphes spécifiques, et seules les DPDU avec des GraphID concordants doivent être considérées comme étant des candidats pour de telles liaisons. Si plusieurs DPDU sur la file d'attente de messages sont des candidats pour une liaison donnée, la DPDU est sélectionnée par priorité comme décrit en 9.1.8.5.

9.4.3.7.2 Sémantique

Le Tableau 180 spécifie les champs pour dlmo.Link.

Tableau 180 – Champs de dlmo.Link

Nom de champ	Codage de champ
* Index	Type: ExtDLUint (utilisé comme un indice)
SuperframeIndex (référence à une entrée de dlmo.Superframe)	Type: ExtDLUint
Type (voir Tableau 182 et le texte associé)	Type: Unsigned8
Template1 (référence de dlmo.TsTemplate au modèle principal)	Type: ExtDLUint
Template2 (référence de dlmo.TsTemplate au modèle secondaire)	Type: ExtDLUint ou null
NeighborType	Type: Unsigned2 Plage valide: 0..2
GraphType	Type: Unsigned2 Plage valide: 0..2
SchedType	Type: Unsigned2
ChType	Type: Unsigned1
PriorityType	Type: Unsigned1
Neighbor (identifier le voisin ou le groupe)	Type: ExtDLUint ou null
GraphID (identité de 12 bits du graphe avec accès exclusif ou hiérarchisé par ordre de priorité à la liaison)	Type: ExtDLUint ou null
Schedule (programme de liaisons; voir Tableau 184)	Type: Voir Tableau 184
ChOffset (sélectionner la voie en fonction du décalage par rapport au modèle de saut de supertrames)	Type: Unsigned8 ou null
Priority (priorité de liaison)	Type: Unsigned8 ou null Plage valide: 0 x 0000 xxxx

Le Tableau 181 montre la structure de dlmo.Link. Les champs ExtDLUint y sont montrés comme étant des octets seuls.

Tableau 181 – Structure de dlmo.Link

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1..2	* Index							
1	SuperframeIndex (plage 1..127)							
1	Type							
1	Template1 (plage 1..127)							
0 ou 1	Template2 (plage 1..127)							
1	NeighborType		GraphType		SchedType		ChType	PriorityType
0..2	Neighbor							
0..2	GraphID							
1..4	Schedule (voir Tableau 184)							
0 ou 1	ChOffset							
0 ou 1	Priority							

Les éléments comprennent:

- dlmo.SuperframeIndex. Indique la référence de supertrame pour la liaison.
- dlmo.Link[ ].Type. Indique comment la liaison est configurée pour l'émission et/ou la réception, et/ou la découverte de voisins. Voir Tableau 182.
- dlmo.Link[ ].Template1. Modèle d'intervalle de temps principal. Voir 9.4.3.3 pour un débat relatif aux modèles.
- dlmo.Link[ ].Template2. Modèle d'intervalle de temps secondaire, pour des intervalles de temps d'émission/réception (T/R) uniquement, en combinaison avec d'autres sélections de liaisons. Utiliser Template2 comme modèle de récepteur de transaction, s'il n'y a aucune DPDU dans la file d'attente pour le modèle principal. Template2 n'est émis et n'a de sens que pour les liaisons TRx, c'est-à-dire les liaisons où les bits 6 et 7 Link[ ].Type ont tous les deux une valeur de 1. Voir 9.1.8.5.
- dlmo.Link[ ].NeighborType et dlmo.Link[ ].Neighbor. Un voisin est désigné pour des liaisons d'émission, et peut être désigné pour des liaisons de réception en duodiffusion/diffusion N. Voir 9.4.3.4 pour un débat relatif aux voisins. Lorsqu'un voisin est désigné dans une liaison, il peut référencer soit un indice dlmo.Neighbor, soit un groupe (dlmo.Neighbor[ ].GroupCode).
  - Si dlmo.Link[ ].NeighborType=0, l'attribut dlmo.Link[ ].Neighbor est null (vide) et n'est pas émis.
  - Si dlmo.Link[ ].NeighborType=1, l'attribut dlmo.Link[ ].Neighbor désigne un indice dans l'attribut dlmo.Neighbor.
  - Si dlmo.Link[ ].NeighborType=2, l'attribut dlmo.Link[ ].Neighbor désigne un groupe.
- dlmo.Link[ ].GraphType, dlmo.Link[ ].GraphID. Les DPDU suivant un graphe particulier peuvent recevoir un accès exclusif ou prioritaire à certaines liaisons d'émission. Ces champs, lorsqu'ils ont été configurés comme tels, limitent l'accès des liaisons à certains graphes, reliant donc la liaison à un flux particulier de communication à travers le sous-réseau D. Lorsque GraphType est laissé vide, la liaison d'émission est disponible à toute DPDU qui est acheminée par l'intermédiaire du voisin désigné de la liaison. Lorsque GraphType est utilisé, un graphe particulier reçoit un accès exclusif ou prioritaire à la liaison.
  - Si GraphType=0, l'élément GraphID est null et n'est pas émis.
  - Si GraphType=1, la liaison est désignée pour un usage exclusif par un graphe particulier. L'accès à la liaison doit être limité aux DPDU suivant le graphe en question.
  - Si GraphType=2, les DPDU avec un ID de graphe concordant reçoivent un accès prioritaire. Il convient que les DPDU suivant le graphe en question reçoivent une priorité sur les autres DPDU lorsque la liaison est utilisée.
- dlmo.Link[ ].SchedType, dlmo.Link[ ].Schedule. Indique la (les) position(s) d'intervalle de temps de la liaison au sein de chaque cycle de supertrame. Le programme peut désigner un décalage fixe, un ensemble répétitif avec un intervalle fixe, une plage, ou une carte de bits, comme suit:

  0: décalage uniquement;

  1: décalage et intervalle;

  2: plage;

  3: carte de bits;
- dlmo.Link[ ].ChType, dlmo.Link[ ].ChOffset. Indique comment la voie de la liaison est sélectionnée.
  - Si dlmo.Link[ ].ChType=0, l'attribut dlmo.Link[ ].ChOffset est null et n'est pas émis. Suivre simplement le modèle de saut de voie de ligne de base de la trame.
  - Si dlmo.Link[ ].ChType=1, ajouter dlmo.Link[ ].ChOffset au courant dlmo.Superframe[ ].ChOffset, modulo de la taille effective du modèle de saut de voie (après avoir rendu compte des voies exclues) des supertrames, et sélectionner la voie en conséquence.

- dlmo.Link[ ].PriorityType, dlmo.Link[ ].Priority. Indique comment la priorité des liaisons est établie. Les priorités de liaison sont décrites fonctionnellement en 9.1.8.5.
  - Si dlmo.Link[ ].PriorityType=false, l'attribut dlmo.Link[ ].Priority est null et n'est pas émis. Pour les liaisons d'émission, utiliser la priorité dlmo.LinkPriorityXmit. Pour les liaisons de réception, utiliser la priorité dlmo.LinkPriorityRcv.
  - Si dlmo.Link[ ].PriorityType=true, utiliser l'attribut dlmo.Link[ ].Priority. Si la liaison est à la fois une liaison d'émission et une liaison de réception, utiliser dlmo.Link[ ].Priority pour l'émission et dlmo.LinkPriorityRcv pour la réception.

Le Tableau 182 montre la structure du champ dlmo.Link[ ].Type.

Tableau 182 – Structure de dlmo.Link[ ].Type

Octet	Bits							
	7	6	5	4	3	2	1	0
1	Transmit	Receive	Exponential backoff	Idle	Discovery: Valeurs nommées: 0: aucune 1: annonce 2: réservé 3: sollicitation		JoinResponse: liaison utilisée pour envoyer une réponse de rattachement à un voisin	SelectiveAllowed

Les types de liaison sont définis comme suit:

- bit 7: Transmit (T=true). Indique l'émission de charge utile;
- bit 6: Receive (R=true);
- bit 5: Exponential backoff (B=true). Indique s'il convient que l'initiateur de la transaction applique les règles de repli exponentiel pour les répétitions de tentative (partagé), versus l'utilisation de l'intervalle de temps sans égard pour le repli exponentiel (non partagé). Voir 9.1.8.2;
- bit 4: Idle (I=true). Lorsque la valeur est "true", la liaison doit être inactive à moins qu'elle ne soit temporairement activée conjointement avec l'émission d'un sous-en-tête DAUX d'activation; voir 9.3.5.4. Lorsque la valeur est "false", l'activation de liaison ne s'applique pas à la liaison. Un intervalle de temps désigné comme étant inactif peut inclure une référence de voisin. Même sans référence de voisin, il a besoin de Booleans d'émission et de réception positionnés selon les besoins pour se référer au modèle d'intervalle de temps dont il aura besoin lorsqu'il sera activé;
- bits 3..2: Discovery (DD). Configuration d'annonce ou de sollicitation. DD='01' spécifie une seule annonce émise avec la temporisation telle que définie dans le modèle d'intervalle de temps. Si DD='01', il convient que la liaison soit utilisée pour émettre une annonce ou une sollicitation, même s'il n'y a pas de charge utile d'ordre supérieur qui a besoin d'être envoyée. DD='11' distingue une sollicitation issue d'une annonce. DD='10' doit être ignoré.

NOTE DD='10' était utilisé par ISA 100.11a pour une autre forme d'annonce, et n'est ainsi pas disponible pour une affectation future. Son utilisation n'est pas prise en charge par la présente norme;

- bit 1: JoinResponse (J=true). Lorsqu'un routeur est un proxy pour une demande de rattachement pour un voisin immédiat, il recevra finalement un message issu du gestionnaire de système à transmettre à la DLE qui se rattache. Le routeur transmet ces messages en utilisant les liaisons qui sont marquées avec JoinResponse=true. Une réponse de rattachement dans la file d'attente de messages peut être identifiée par un en-tête de DPDU avec une adresse de destination EUI64Address. Un intervalle de temps désigné comme prenant en charge une réponse de rattachement peut inclure une référence de voisin, permettant ainsi que la liaison soit également utilisée pour le trafic de sous-réseau D régulier. Même sans référence de voisin, il a besoin que le boolean d'émission soit mis;

- bit 0: SelectiveAllowed (S=true). La DLE peut, sans instruction du gestionnaire de système, traiter de façon autonome et sélective les liaisons d'émission comme étant inactives si elles se produisent sur certaines voies radio avec des antécédents de connectivité médiocre. Il s'agit d'une forme d'utilisation de voie sélective, qui est décrite en 9.1.7.2.4. La DLE peut sauter les liaisons se produisant sur des voies qu'elle juge de façon autonome comme étant problématiques en raison d'antécédents de connectivité médiocre, potentiellement avec la granularité d'une voie spécifique utilisée pour la communication avec un voisin spécifique. De cette manière, la DLE peut économiser l'énergie et réduire les interférences inutiles avec d'autres utilisateurs du spectre. Cependant, la DLE ne doit pas sauter les liaisons de cette façon lorsque la liaison est marquée avec SelectiveAllowed=0.

Le Tableau 183 montre des combinaisons autorisées de bits dans la représentation de dlmo.Link[ ].Type. Les bits montrés comme étant X indiquent qu'une valeur "false" (0) ou "true" (1) est autorisée. Par exemple, plusieurs combinaisons impliquant Transmit=true présentent un X pour le repli exponentiel ("Exponential backoff"), indiquant que de telles liaisons pourraient être configurées comme étant partagées ou non.

Tableau 183 – Combinaisons autorisées de dlmo.Link[ ].Type

Combinaison TRBI DDJS	Description
1X10 001X	Réponse de rattachement
10XX 000X	Emission, pas d'annonce
11XX 000X	Emission/réception, pas d'annonce
10X0 010X	Emission, annonce
0000 0100	Annonce dédiée
0000 1100	Sollicitation
010X 0000	Réception

Une combinaison de liaisons d'émission/réception est essentiellement une représentation compressée d'une liaison d'émission et d'une liaison de réception distincte. Si au moins une DPDU sortante dans la file d'attente concorde avec la liaison, elle doit être traitée comme étant une liaison d'émission en utilisant le modèle d'intervalle de temps principal. Autrement, elle doit être traitée comme étant une liaison de réception en utilisant le modèle d'intervalle de temps secondaire, avec la priorité dlmo.LinkPriorityRcv.

Des modèles de saut de voie, des périodes de supertrame et des intervalles de liaison peuvent être configurés afin que seules certaines voies radio dans la séquence de saut de voie soient réellement utilisées. Par exemple, si une liaison se répète tous les 20 intervalles de temps, et le modèle de saut de voie inclut 15 voies, alors seules trois voies seront réellement utilisées par la liaison. Pour éviter de tels scénarios, l'intervalle de liaison et le modèle de saut de voie peuvent être configurés de façon à disposer d'un plus grand commun diviseur (PGCD) égal à un (1).

Le Tableau 184 spécifie les différents types de programmes pour une liaison donnée. Les liaisons dans une supertrame sont indexées en fonction du décalage d'intervalle de temps dans chaque cycle, le premier intervalle de temps dans chaque cycle ayant un décalage zéro.

Tableau 184 – Valeurs pour dlmo.Link[ ].Schedule

Valeur pour dlmo.Link[ ].Schedule	Codage d'élément	Description
0 = décalage uniquement	ExtDLUint (décalage)	La liaison se produit une fois à une position d'intervalle de temps fixe dans chaque cycle de supertrame
1 = décalage et intervalle	ExtDLUint, ExtDLUint (décalage, intervalle)	La liaison se produit plusieurs fois dans chaque cycle, d'abord au décalage donné puis en se répétant à un certain intervalle jusqu'à la fin du cycle. Les valeurs sont spécifiées en nombre d'intervalles de temps
2 = plage	ExtDLUint, ExtDLUint (premier, dernier)	La liaison se produit à une plage d'intervalles de temps dans chaque cycle de supertrame, en commençant par le décalage donné par la première valeur et en continuant jusqu'au décalage donné par la seconde valeur
3 = carte de bits	BooleanArray32	La carte de bits couvre les 32 premiers intervalles de temps dans chaque cycle de supertrame. La liaison se produit dans des intervalles de temps avec une valeur "true" correspondante dans la matrice. En suivant les conventions LSB, la matrice est émise dans des messages de DMAP avec les indices 7..0 émis les premiers et les indices 31..24 émis les derniers

9.4.3.8 dlmo.Route

9.4.3.8.1 Généralités

L'attribut dlmo.Route est un ensemble OctetString indexé qui contient des chemins. L'attribut dlmo.Route décrit les chemins disponibles pour les DPDU. Lorsqu'une DSDU descend la pile de protocoles issue de la NL, elle reçoit une adresse de destination finale, accompagnée d'un ID de contrat et d'une classe de priorités. La DLE met en correspondance l'ID de contrat et l'adresse de destination avec un chemin, en se basant sur des consultations de table. La classe de priorités issue de la NL est simplement copiée dans l'en-tête de la DPDU sans être prise en considération dans la sélection du chemin.

Le gestionnaire de système insère, met à jour ou supprime des chemins en envoyant au DMAP un chemin, accompagné d'un indice unique et (si sélectionné) d'un temps TAI de basculement.

Pour une description de cette sélection de chemin, voir 9.1.6.5.

9.4.3.8.2 Sémantique

Le Tableau 185 spécifie les champs pour dlmo.Route.

Tableau 185 – Champs de dlmo.Route

Nom de champ	Codage de champ
* Index	Type: ExtDLUint (utilisé comme un indice)
Size [taille (nombre d'entrées) de l'attribut Route]	Type: Unsigned4 Plage valide: 1..15
Alternative	Type: Unsigned2
Reserved (alignement d'octets)	Type: Unsigned2=0
ForwardLimit (valeur d'initialisation pour la limite de transmission en DPDU qui utilisent ce chemin)	Type: Unsigned8
Route (série de destinations de routage; si le bit de poids fort d'entrée est 0, cela représente une adresse de monodiffusion; si les bits de poids fort d'entrée sont 0x 1010, cela représente un graphe)	Type: SEQUENCE OF Unsigned16 (SIZE (size))
Selector (voir texte)	Type: Unsigned16 ou null
SrcAddr (voir texte)	Type: ExtDLUint ou null

Le Tableau 186 montre la structure de dlmo.Route.

Tableau 186 – Structure de dlmo.Route

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1..2	* Index							
1	Size				Alternative		Reserved = 0	
1	ForwardLimit							
2	$Route_0$							
...	...							
2	$Route_{Size-1}$							
0 ou 2	Selector							
0..2	SrcAddr							

L'attribut dlmo.Route[ ].Selector dépend de la valeur de réglage de dlmo.Route[ ].Alternative:

- lorsque dlmo.Route[ ].Alternative=0, sélectionner ce chemin si dlmo.Route[ ].Selector concorde avec le ContractID et dlmo.Route[ ].SrcAddr concorde avec le champ SrcAddr (adresse source) dans le sous-en-tête DADDR. Cette alternative ne doit pas être utilisée, sauf si la DLE est un routeur dorsal. Si dlmo.Route[ ].Alternative<>0, l'attribut dlmo.Route[ ].SrcAddr est null (vide) et ne doit pas être émis;
- lorsque dlmo.Route[ ].Alternative=1, sélectionner ce chemin si dlmo.Route[ ].Selector concorde avec ContractID;
- lorsque dlmo.Route[ ].Alternative=2, sélectionner ce chemin si dlmo.Route[ ].Selector concorde avec l'adresse de destination;
- lorsque dlmo.Route[ ].Alternative=3, utiliser ce chemin comme valeur par défaut. L'attribut dlmo.Route[ ].Selector est null (vide) et ne doit pas être émis.

L'attribut dlmo.Route[ ].Alternative doit être appliqué dans l'ordre, en donnant aux alternatives ayant les bas numéros la préséance sur les alternatives ayant les numéros plus élevés. Il convient qu'il n'y ait pas plus d'une entrée dlmo.Route par combinaison de ContractID/SrcAddr (Alternative=0), pas plus d'une entrée par ContractID (Alternative=1), pas plus d'une entrée par adresse de destination (Alternative=2), et pas plus d'une valeur par

défaut (Alternative=3). S'il y a des doublons, l'entrée concordante ayant l'indice le plus faible doit être sélectionnée.

9.4.3.9 dlmo.NeighborDiag

9.4.3.9.1 Généralités

L'attribut dlmo.NeighborDiag est un ensemble OctetString indexé qui contient les diagnostics pour un jeu de voisins. L'attribut est en lecture seule, avec des rangées créées selon les besoins par la DLE.

Chaque entrée de NeighborDiag comporte une matrice d'un ou deux OctetString, chaque entrée correspondant à un voisin différent.

Les entrées de NeighborDiag sont instanciées par le gestionnaire de système, en mettant les bits de dlmo.Neighbor[ ].DiagLevel à des valeurs différentes de zéro. Si et seulement si Bit0=1, les diagnostics de résumés doivent être recueillis pour le voisin, consolidés à travers toutes les voies. Si et seulement si Bit1=1, les diagnostics d'horloge détaillés doivent être recueillis pour le voisin, consolidés à travers toutes les voies radioélectriques.

NOTE Les diagnostics de voie individuels sont recueillis par l'intermédiaire de l'attribut dlmo.ChannelDiag.

Les diagnostics incluent des compteurs et des niveaux, qui sont accumulés selon la description en 9.1.15.3. En général, les compteurs sont incrémentés de 1 au cours des transactions réussies ou non réussies, alors que RSSI (intensité de signal) et RSQI (qualité de signal) sont des niveaux qui sont accumulés sous la forme de moyennes mobiles exponentielles.

NeighborDiag est rapporté de trois manières générales:

- par l'intermédiaire du HRCO, le gestionnaire de système peut configurer la DLE pour rapporter périodiquement NeighborDiag, par exemple toutes les 30 min. Après chaque rapport de ce type, sur une base par entrée, les comptes de NeighborDiag doivent être réinitialisés à zéro. Les niveaux doivent utiliser la valeur courante comme un point de départ pour la prochaine période;
- le gestionnaire de système peut lire (interroger) NeighborDiag comme un attribut en lecture seule sur une base entrée par entrée. Comme dans un rapport de HRCO, les comptes doivent être réinitialisés à zéro après une lecture;
- la DLE peut en outre être configurée, par l'intermédiaire de l'attribut dlmo.AlertPolicy, pour rapporter des informations de NeighborDiag lorsque des valeurs de diagnostics dépassent un seuil. Seule la rangée déclenchant l'alerte est rapportée. Aucune valeur n'est réinitialisée à zéro.

En général, dans la présente norme, la capacité des métadonnées d'un OctetString indexé est rapportée comme étant le nombre de rangées. Sachant que les rangées dans NeighborDiag peuvent avoir des tailles substantiellement variables, les métadonnées pour NeighborDiag (DiagMeta) doivent être rapportées en capacité mémoire en octets pour les OctetString, avec la convention selon laquelle chaque champ ExtDLUint est supposé consommer deux octets. Une DLE doit avoir la capacité pour des diagnostics de résumés (dlmo.NeighborDiag[ ].Summary) pour au moins la moitié de sa capacité de voisins telle qu'indiquée par dlmo.NeighborMeta.Capacity ou pour au moins deux voisins, la valeur la plus grande étant retenue.

9.4.3.9.2 Sémantique

Chaque entrée de NeighborDiag inclut trois OctetString, un chacun pour les diagnostics Summary et ClockDetail. Un OctetString de longueur zéro indique que le diagnostic n'est pas accumulé. Le Tableau 187 spécifie les champs pour dlmo.NeighborDiag.

Tableau 187 – Champs de dlmo.NeighborDiag

Nom de champ	Codage de champ
* Index	Type: ExtDLUint (adresse de voisin, utilisée comme un indice)
Summary	Type: OctetString
ClockDetail	Type: OctetString

Le Tableau 188 spécifie les champs au sein de l'OctetString "Summary" de diagnostic.

Tableau 188 – Champs de l'OctetString "Summary" de diagnostic

Nom de champ	Codage de champ
RSSI (niveau)	Type: Integer8
RSQI (niveau)	Type: Unsigned8
RxDPDU (compte)	Type: ExtDLUint
TxSuccessful (compte)	Type: ExtDLUint
TxFailed (compte)	Type: ExtDLUint
TxCCA_Backoff (compte)	Type: ExtDLUint
TxNAK (compte)	Type: ExtDLUint
ClockSigma (niveau)	Type: Integer16

Le Tableau 189 spécifie la structure des OctetString "Summary" de diagnostic.

Tableau 189 – Structure de l'OctetString "Summary" de diagnostic

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	RSSI							
1	RSQI							
1 ou 2	RxDPDU							
1 ou 2	TxSuccessful							
1 ou 2	TxFailed							
1 ou 2	TxCCA_Backoff							
1 ou 2	TxNAK							
2	ClockSigma							

Les champs comprennent:

- RSSI (intensité de signal): voir 9.1.15.2 pour un débat relatif aux unités de RSSI. RSSI est cumulé sous la forme d'une moyenne mobile exponentielle; voir 9.1.15.3;
- RSQI (qualité de signal): voir 9.3.5.5 et 9.1.15.2 pour un débat relatif aux unités de RSQI. RSQI est cumulé sous la forme d'une moyenne mobile exponentielle; voir 9.1.15.3;
- RxDPDU: compte des DPDU Data valides reçues en provenance du voisin, à l'exclusion des DPDU sans charge utile de DSDU (mais également des DPDU ACK/NAK);
- TxSuccessful: compte des émissions en monodiffusion réussies vers le voisin, où un DPDU ACK a été reçu en réponse;

- TxFailed: compte des émissions en monodiffusion de DPDU, où un DPDU ACK/NAK était attendu mais n'a pas été reçu en réponse;
- TxCCA_Backoff: compte des émissions en monodiffusion qui ont été abandonnées en raison de la CCA. Ces émissions abandonnées ne sont pas incluses dans TxFailed;
- TxNAK: compte des NAK reçus, non inclus dans TxFailed;
- ClockSigma: estimation grossière, au sein d'un écart type, des récentes corrections d'horloge, en unités de 2^{-20} s. Une valeur de 1 sigma rend compte d'approximativement 68 % des corrections d'horloge. ClockSigma est réinitialisé à zéro chaque fois que les compteurs sont réinitialisés.

NOTE 1 Voir 9.1.15.3 pour le comportement des compteurs.

Si la DLE traite de façon autonome une liaison d'émission comme étant inactive, telle que décrite en 9.1.7.2.4, de telles liaisons sautées ne doivent pas être comptées dans les diagnostics de voisin. Cependant, ces liaisons sautées sont reflétées dans le diagnostic de voie, tel que décrit en 9.4.2.27.

Le Tableau 190 spécifie les champs au sein de l'OctetString ClockDetail de diagnostic.

Tableau 190 – Champs de l'OctetString ClockDetail de diagnostic

Nom de champ	Codage de champ
ClockBias (niveau, signé)	Type: Integer16
ClockCount (compte)	Type: ExtDLUint
ClockTimeout (compte)	Type: ExtDLUint
ClockOutliers (compte)	Type: ExtDLUint

Le Tableau 191 spécifie la structure de l'OctetString ClockDetail de diagnostic, avec les champs ExtDLUint montrés sous forme d'un seul octet.

Tableau 191 – Structure de l'OctetString ClockDetail de diagnostic

Octets	Bits							
	7	6	5	4	3	2	1	0
2	ClockBias							
1 ou 2	ClockCount							
1 ou 2	ClockTimeout							
1 ou 2	ClockOutliers							

Si le voisin est une source d'horloge de DL préférentielle, telle qu'indiquée par IncludeClock=1, il convient que la DLE soit configurée pour accumuler les champs de ClockDetail.

ClockSigma, ClockBias, et ClockOutliers se rapportent tous à des corrections d'horloge. Si le voisin est une source d'horloge de DL, ces valeurs se rapportent à des corrections d'horloge reçues en provenance de la source d'horloge de DL. Si le voisin n'est pas source d'horloge de DL, ces valeurs se relient aux corrections d'horloge envoyées vers le voisin de la DLE par l'intermédiaire de la DPDU ACK/NAK.

Les champs comprennent:

- ClockBias: une moyenne mobile exponentielle (EMA) de correction d'horloge, en unités de 2^{-20} s, y compris le signe, avec un facteur de lissage de 1 %. Voir 9.1.15.3 pour un débat relatif à l'EMA.

NOTE 2 Si cette valeur est sensiblement différente de zéro, elle indique que l'horloge est biaisée par rapport à la source d'horloge distante;

- ClockCount: compte des mises à jour d'horloge reçues en provenance du voisin ou émises vers celui-ci;
- ClockTimeout: compte des événements de temporisation d'horloge;
- ClockOutliers: compte estimé des corrections d'horloge de plus de trois écarts types selon ClockSigma.

9.5 Méthodes de DLE

9.5.1 Méthode pour le basculement synchronisé des attributs de DLE

Une méthode Scheduled_Write, avec MethodID=1, est fournie pour positionner un attribut à un temps TAI spécifique. Elle suit exactement le modèle donné dans le Tableau J.1.

9.5.2 Méthodes pour accéder aux attributs OctetString indexés

Les diverses méthodes dans la DLE se rapportent à l'écriture, à la lecture, et à la suppression d'attributs OctetString indexés. Ces méthodes sont généralement basées sur les modèles fournis en Annexe J.

Tous les attributs OctetString indexés dans la DL sont indexés par un seul nombre entier codé comme étant une ExtDLUint (voir 9.3.2.2). Suivant la convention des méthodes de modèle, ces indices sont dupliqués dans les arguments d'entrée. Par exemple, la méthode Write_Row inclut un indice comme argument d'entrée même si l'indice est également transporté au sein de l'OctetString qui constitue la nouvelle entrée.

Le Tableau 192 spécifie la méthode Read_Row.

Tableau 192 – Méthode Read_Row

Nom de méthode	ID de méthode	Description de la méthode			
Read_Row	2	Méthode pour lire la valeur d'une seule rangée d'un attribut OctetString indexé dont les données sont visualisées sous la forme d'un tableau d'informations			
	Arguments d'entrée				
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**	
	1	Attribute_ID	Unsigned16	L'ID d'attribut dans le DLMO auquel la présente méthode est appliquée	
	2	Index	Unsigned16	Le champ * Index dans l'attribut pour accéder à une rangée particulière	
	Arguments de sortie				
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**	
	1	Data_Value	OctetString	Une chaîne d'octets qui contient le contenu de la rangée. Si la rangée est vide, l'OctetString ne doit contenir que l'Index, codé comme ExtDLUint	

Le Tableau 193 spécifie la méthode Write_Row.

Tableau 193 – Méthode Write_Row

Nom de méthode	ID de méthode	Description de la méthode		
Write_Row	3	Méthode pour établir/modifier la valeur d'une seule rangée d'un attribut OctetString indexé dont les données sont visualisées sous la forme d'un tableau d'informations		
	Arguments d'entrée			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	Attribute_ID	Unsigned16	L'ID d'attribut dans le DLMO auquel cette méthode est appliquée, tel que déterminé par l'index ordinal de l'attribut dans la définition du DLMO
	2	Scheduled_TAI_Time	Unsigned32	Temps TAI en secondes auquel il convient que la valeur soit écrite dans la rangée de l'attribut structuré. Si le temps se situe dans le passé, par rapport au sens du temps de l'appareil de réception, l'écriture doit être effectuée immédiatement
	3	Index	Unsigned16	Le champ * Index dans l'attribut pour accéder à une rangée particulière
	4	Data_Value	OctetString	Une chaîne d'octets qui contient le nouveau contenu de la rangée. Si la rangée de DLMO n'est pas peuplée, il est créé une nouvelle rangée contenant l'OctetString si de la mémoire est disponible. Si la rangée de DLMO existe déjà, son contenu est remplacé par l'OctetString. Si l'OctetString est null (vide), alors la rangée doit être supprimée
	Arguments de sortie			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	Aucun			

Le Tableau 194 spécifie la méthode Write_Row_Now. Elle est identique à la méthode Write_Row, sans l'argument Scheduled_TAI_Time. Elle a l'effet d'écrire une rangée OctetString indexé immédiatement à la réception.

Tableau 194 – Méthode Write_Row_Now

<table>
<tr><th>Nom de méthode</th><th>ID de méthode</th><th colspan="3">Description de la méthode</th></tr>
<tr><td rowspan="9">Write_Row_Now</td><td>4</td><td colspan="3">Méthode pour établir/modifier la valeur d'une seule rangée d'un attribut OctetString indexé dont les données sont visualisées sous la forme d'un tableau d'informations</td></tr>
<tr><td colspan="4">Arguments d'entrée</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td>1</td><td>Attribute_ID</td><td>Unsigned16</td><td>L'ID d'attribut dans le DLMO auquel cette méthode est appliquée, tel que déterminé par l'index ordinal de l'attribut dans la définition du DLMO</td></tr>
<tr><td>2</td><td>Index</td><td>Unsigned16</td><td>Le champ * Index dans l'attribut pour accéder à une rangée particulière</td></tr>
<tr><td>3</td><td>Data_Value</td><td>OctetString</td><td>Une chaîne d'octets qui contient le nouveau contenu de la rangée. Si la rangée de DLMO n'est pas peuplée, il est créé une nouvelle rangée contenant l'OctetString si de la mémoire est disponible. Si la rangée de DLMO existe déjà, son contenu est remplacé par l'OctetString. Si l'OctetString est null (vide), alors la rangée doit être supprimée</td></tr>
<tr><td colspan="4">Arguments de sortie</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td colspan="4">Aucun</td></tr>
</table>

9.6 Alertes de DL

9.6.1 Alerte DL_Connectivity

Les diagnostics des performances de DLE sont accumulés dans les attributs dlmo.NeighborDiag pour les diagnostics par voisin, et dlmo.ChannelDiag pour les diagnostics par voie. Normalement, le gestionnaire de système configure le HRCO pour rapporter ces diagnostics périodiquement, et la DLE réinitialise automatiquement les compteurs de diagnostics chaque fois que ces attributs sont ainsi rapportés. Entre de tels rapports, les diagnostics peuvent indiquer un problème qui a besoin d'être immédiatement rapporté au gestionnaire de système. L'alerte DL_Connectivity fournit le mécanisme permettant à la DLE de rapporter de tels problèmes, et l'attribut dlmo.AlertPolicy permet au gestionnaire de système d'établir des seuils pour la production de tels rapports.

L'attribut dlmo.AlertPolicy active/désactive l'alerte de DL_Connectivity et fournit des seuils pour commander si, oui ou non, des alertes sont rapportées. L'attribut dlmo.AlertPolicy est un OctetString contenant des champs tels que montrés dans le Tableau 195.

Tableau 195 – Champs de dlmo.AlertPolicy

Nom de champ	Codage de champ
Descriptor (active ou désactive l'alerte DL_Connectivity)	Type: Alert report descriptor Par défaut: Disabled=true Par défaut: Priority=0
NeiMinUnicast (nombre minimal de transactions en monodiffusion nécessaires pour un rapport de voisin)	Type: ExtDLUint
NeiErrThresh (rapporter le diagnostic de voisin si le taux d'erreur en pourcentage atteint ce seuil)	Type: Unsigned8
ChanMinUnicast (nombre minimal de transactions en monodiffusion sur une voie nécessaires comme condition préalable du déclenchement d'une alerte)	Type: ExtDLUint
NoAckThresh (rapporter ChannelDiag si une valeur NoAck est supérieure à ce seuil)	Type: Unsigned8
CCABackoffThresh (rapporter ChannelDiag si une valeur CCABackoff est supérieure à ce seuil)	Type: Unsigned8

Le Tableau 196 spécifie la structure de l'OctetString dlmo.AlertPolicy.

Tableau 196 – Structure de l'OctetString dlmo.AlertPolicy

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
2	Descriptor							
1 ou 2	NeiMinUnicast							
1	NeiErrThresh							
1 ou 2	ChanMinUnicast							
1	NoAckThresh							
1	CCABackoffThresh							

Les champs comprennent:

- l'attribut dlmo.AlertPolicy.Descriptor détermine si, oui ou non, l'alerte DL_Connectivity est activée. Par défaut, l'alerte DL_Connectivity est désactivée jusqu'à ce que le gestionnaire de système l'active en peuplant cet attribut avec les seuils appropriés. Lorsque Disabled=true, tous les autres champs de dlmo.AlertPolicy sont sans signification et sont ignorés;
- l'attribut dlmo.AlertPolicy.NeiMinUnicast établit un nombre minimal de transactions tentées en monodiffusion avant qu'un taux d'erreur soit considéré comme étant significatif. Le compte de transactions tentées en monodiffusion pour un voisin est la somme des champs de l'attribut dlmo.NeighborDiag

  TxSuccessful + TxFailed + TxCCA_Backoff + TxNAK.

  Si cette somme est inférieure à NeighborTxMinReport, ne pas envoyer d'alerte DL_Connectivity pour le voisin;
- l'attribut dlmo.AlertPolicy.NeiErrThresh établit le seuil pour rapporter une alerte DL_Connectivity pour un voisin. Le taux d'erreur en pourcentage est calculé comme étant:

  (TxFailed + TxCCA_Backoff + TxNAK) x 100 /
  (TxSuccessful + TxFailed + TxCCA_Backoff + TxNAK)

  Si cette valeur est supérieure à NeiErrThresh, il convient que les diagnostics pour le voisin soient rapportés en utilisant l'alerte DL_Connectivity, à moins qu'il n'y ait un nombre insuffisant de transactions en monodiffusion vers le voisin ou à moins que la même alerte n'ait été rapportée récemment;

- l'attribut dlmo.AlertPolicy.ChanMinUnicast est similaire à NeiMinUnicast. Les compteurs sous-tendant dlmo.ChannelDiag ne sont pas exposés, mais un compte des transactions tentées en monodiffusion est implicite dans les ratios rapportés;
- les attributs dlmo.AlertPolicy.NoAckThresh et dlmo.AlertPolicy.CCABackoffThresh fournissent des seuils pour la production de rapports. Sachant que les valeurs rapportées par dlmo.ChannelDiag sont des ratios, les valeurs rapportées sont simplement comparées aux seuils. Si la valeur excède les seuils, il convient que dlmo.ChannelDiag soit rapporté par l'intermédiaire de l'alerte DL_Connectivity, à moins que l'exigence relative à ChanMinUnicast ne soit pas respectée ou à moins que la même alerte n'ait été rapportée récemment.

Le gestionnaire de système peut répondre à l'alerte de connectivité de DL en recueillant les diagnostics afin de caractériser plus complètement la situation. En variante, en particulier si une topologie modifiée est facilement obtenue, le gestionnaire de système peut simplement reconfigurer la topologie du sous-réseau D.

Le Tableau 197 montre la structure de l'alerte DL_Connectivity.

Tableau 197 – Alerte DL_Connectivity

Nom du type d'objet normalisé: DL management object (DLMO, objet de gestion de DL)					
Identificateur du type d'objet normalisé: 124					
Description de l'alerte: connectivité médiocre de voisin					
Classe d'alertes (Enumerated: alarme ou événement)	**Catégorie d'alertes (Enumerated: diagnostic d'appareil, diagnostic de comm., sécurité ou processus)**	**Type d'alerte (Enumerated: en fonction de la catégorie d'alertes)**	**Priorité d'alerte**	**Type de données de la valeur**	**Description de la valeur incluse avec l'alerte**
Evénement	Comm	0 = DL_Connectivity	Medium	Type: DL16Address	Voir Tableau 187

Le format de l'OctetString émis avec l'alerte DL_Connectivity est montré dans le Tableau 198. Il s'agit simplement du numéro d'attribut pour dlmo.ChannelDiag (48) ou pour dlmo.NeighborDiag (46), suivi d'un OctetString contenant les données de diagnostic issues de l'attribut en question. Dans le cas de ChannelDiag, l'attribut entier est émis. Dans le cas de NeighborDiag, seule la rangée qui a déclenché l'alerte est émise, avec l'adresse de voisin spécifiée au sein de la rangée identifiant le voisin.

Tableau 198 – OctetString d'alerte DL_Connectivity

Octets	**Bits**							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	AttributeNumber (Unsigned8)							
N	Attribute (OctetString)							

9.6.2 Alerte NeighborDiscovery

Comme décrit en 9.4.2.24, l'alerte NeighborDiscovery fournit un mécanisme permettant à la DLE de rapporter le contenu de l'OctetString dans l'attribut dlmo.Candidates.

Le Tableau 199 montre la structure de l'alerte NeighborDiscovery.

Tableau 199 – Alerte NeighborDiscovery

Nom du type d'objet normalisé: DL management object (DLMO, objet de gestion de DL)					
Identificateur du type d'objet normalisé: 124					
Description de l'alerte: alerte de découverte de voisin					
Classe d'alertes (Enumerated: alarme ou événement)	**Catégorie d'alertes (Enumerated: diagnostic d'appareil, diagnostic de comm., sécurité ou processus)**	**Type d'alerte (Enumerated: en fonction de la catégorie d'alertes)**	**Priorité d'alerte**	**Type de données de la valeur**	**Description de la valeur incluse avec l'alerte**
Evénement	Comm	1 = NeighborDiscovery	Medium	Type: OctetString	Une copie exacte de l'OctetString dans dlmo.Candidates; voir 9.4.2.24

10 Couche Réseau

10.1 Généralités

L'Article 10 donne une vue d'ensemble de la fonctionnalité de NL. Il décrit également les services conceptuels que la NL offre à la couche au-dessus d'elle (transport), l'objet de gestion de NL (NLMO) et la structure des NPDU.

NOTE Les formats d'en-tête de NPDU ont été conçus pour la compatibilité avec l'IETF RFC 6282.

Le NL suit la convention gros-boutiste; les champs à plusieurs octets sont documentés et émis avec l'octet de poids fort le premier (car ils sont traités comme une série d'octets par la couche inférieure). Au sein d'un octet, les bits sont documentés en commençant à partir du bit de poids fort (bit 7) à gauche et en continuant jusqu'au bit de poids faible (bit 0) à droite.

Des parties de l'Article 10 présentent des aspects hypothétiques de mise en œuvre comme s'ils étaient soumis à des essais de conformité. Lorsque de tels aspects ne sont pas observables extérieurement, ces spécifications sont strictement hypothétiques. Seuls les aspects observables pouvant être soumis à essai de l'Article 10 sont normatifs.

10.2 Vue d'ensemble des fonctionnalités de la NL

10.2.1 Généralités

La NL dans la présente norme accomplit les fonctions suivantes:

- adressage: une NLE détermine les informations appropriées relatives à l'adresse pour une NPDU;
- conversion d'adresses: la présente norme utilise principalement deux types d'adresses, les adresses DL16Addresses courtes et les adresses IPv6Addresses longues. Les adresses courtes DL16Addresses sont utilisées au sein d'un sous-réseau D pour conserver l'énergie et la largeur de bande. Les ALE, les TLE et les NLE sur des réseaux dorsaux utilisent de longues adresses IPv6Addresses. La NLE est responsable de la conversion entre les divers types d'adresses, par exemple, lorsqu'une NPDU se déplace d'un sous-réseau D vers un réseau dorsal (ou vice versa);
- formats de NPDU: la présente norme prévoit plus d'un format de NPDU pour prendre en charge la conservation de l'énergie et de la largeur de bande (qui favorise les en-têtes courts), une diversité de topologies de réseau, et l'interconnexion de réseaux avec des réseaux dorsaux. La NLE sélectionne un format approprié pour la NPDU en se basant sur des considérations telles que l'adressage, le routage, le niveau de service, etc.;

- fragmentation et réassemblage: la fragmentation et le réassemblage de NPDU se produisent au sein de la NLE. Une NPDU de taille supérieure à la taille maximale de DSDU est fragmentée par la NLE expéditrice au point d'entrée du sous-réseau D. Le réassemblage est accompli par la NLE réceptrice au point de sortie du sous-réseau D;
- routage: la présente norme accomplit le routage à deux niveaux: au sein du réseau dorsal et au sein du sous-réseau D de la maille. La responsabilité du routage au niveau des couches de protocole de NL et de DL incombe aux entités des couches respectives.

10.2.2 Adressage

Les ALE et les TLE dans la présente norme utilisent des adresses IPv6Addresses. Chaque NLE doit avoir une adresse IPv6Address. Si la NLE n'a pas une adresse IPv6Address avant le processus de rattachement de réseau, la NLE doit avoir une telle adresse IPv6Address qui lui est assignée par le gestionnaire de système au cours du processus de rattachement. La NL utilise ces adresses IPv6Addresses, mais ne leur associe aucune autre signification.

Chaque NLE conforme à la présente norme doit également avoir une adresse IPv6Address qui est autoconfigurée par la NLE comme partie intégrante de l'initialisation de sa couche de protocoles. Cette adresse IPv6Address est appelée adresse locale à une liaison de la NLE et est dérivée à partir de l'adresse EUI64Address de la DLE qui lui est associée. Le format de cette adresse IPv6Address est celui d'une adresse en monodiffusion locale à une liaison, telle que définie dans l'IETF RFC 4291:2006, 2.5.6. Le Tableau 200 montre la structure de cette adresse.

Tableau 200 – Structure d'adresse locale à une liaison

10 bits	54 bits	64 bits
11 1111 1010	0	EUI64Address

Lorsque les DPDU sont émises sur un sous-réseau D, l'acheminement des adresses IPv6Addresses consomme de précieuses ressources de largeur de bande et d'énergie d'appareil. Aussi, la présente norme définit-elle des pseudonymes D de 16 bits pour les adresses IPv6Addresses afin que de courts pseudonymes D soient utilisés sur le sous-réseau D. Pour chaque sous-réseau D, une adresse DL16Address unique doit être assignée à chaque DLE au sein du sous-réseau D en question, ainsi qu'à chaque DLE située à l'extérieur du sous-réseau D avec laquelle une DLE au sein du sous-réseau D a un contrat. Cela permet d'utiliser des adresses D courtes dans le sous-réseau D pour représenter toutes les NLE d'origine et de destination.

La portée d'une adresse DL16Address est le sous-réseau D au sein duquel elle a été définie. Ainsi, un appareil particulier peut avoir une adresse D dans le sous-réseau D auquel elle appartient et une adresse D différente dans un sous-réseau D étranger. Lorsqu'une adresse DL16Address est utilisée, elle est transportée dans l'en-tête de la DPDU.

Au cours du processus de rattachement, une NLE pourrait ne pas encore avoir une adresse IPv6Address et sa DLE associée pourrait ne pas avoir une adresse DL16Address. Dans ce cas, les TPDU entre l'appareil se rattachant et le routeur D d'annonce de D doivent utiliser les adresses IPv6Addresses locales à une liaison lorsqu'elles sont nécessaires (par exemple, pour le pseudo-en-tête de TPDU dans les TPDU de rattachement). L'appareil se rattachant et le routeur d'annonce doivent être identifiés en tant que tels en utilisant leurs adresses EUI64Addresses dans les en-têtes de DPDU qui acheminent les messages de rattachement.

Le gestionnaire de système assigne respectivement l'adresse DL16Address et l'adresse IPv6Address de chaque DLE et de chaque NLE, qui fonctionnent dans un WISN selon la présente norme. Ces adresses sont assignées au cours du processus de rattachement au sous-réseau. La NLE spécifiée par la présente norme ne prend en charge que l'adressage en monodiffusion.

NOTE 1 L'adressage en multidiffusion est un sujet pour une future normalisation.

NOTE 2 Les technologies de réseau dorsal et de réseau d'installation ne relèvent pas du domaine d'application de la présente norme. Par conséquent, la présente norme ne spécifie pas la représentation des adresses IPv6Addresses sur un réseau dorsal ou d'installation particulier.

10.2.3 Conversion d'adresse

Sachant que la présente norme utilise des adresses DL16Addresses au sein d'un sous-réseau D, lorsqu'une NPDU se déplace d'un sous-réseau D vers un réseau dorsal (ou vice versa), la NLE du routeur dorsal doit effectuer des conversions entre les adresses DL16Addresses et les adresses IPv6Addresses. La même sorte de conversion doit être accomplie par la NLE d'un point d'extrémité de sous-réseau D.

Tous les appareils dans la présente norme doivent maintenir une table de conversion d'adresses (ATT), telle que montrée dans le Tableau 201.

Tableau 201 – Table de conversion d'adresses (ATT)

Adresse D (DL16Address)	Adresse N (IPv6Address)
N1_16	N1_128
N2_16	N2_128
GW_16	GW_128
BBR_16	BBR_128
SM_16	SM_128

La table de conversion d'adresses est initialisée au cours du processus de rattachement au sous-réseau avec l'adresse DL16Address et avec l'adresse IPv6Address du gestionnaire de système. Ces informations sont une partie intégrante de la composante autre que de sécurité de la réponse de rattachement reçue en provenance du gestionnaire de système, telles que décrites en 6.3.9.2.

La table de conversion d'adresses doit être mise à jour par la NLE source chaque fois qu'une session de communication est établie avec une nouvelle NLE de destination. Les sessions de communication sont décrites en 6.3.11.2.5.2. L'adresse DL16Address et l'adresse IPv6Address de la destination NLE de destination et sa DLE associée sont stockées dans la table de conversion d'adresses à l'achèvement réussi du processus d'établissement de session. Le processus d'établissement de session est décrit en 7.5. S'il est mis fin à une session pour une raison quelconque, toute entrée associée à l'appareil de destination doit être supprimée.

Une NLE maintient des entrées dans sa table ATT pour d'autres NLE avec lesquelles elle communique; ces autres NLE pouvant soit appartenir au même sous-réseau D que la première NLE, soit avoir une adresse DL16Address dans le même sous-réseau D.

Au sein d'un sous-réseau D particulier, une NLE (qu'elle soit locale ou distante) doit avoir une seule adresse DL16Address. Ainsi, la table ATT peut être utilisée par la NLE pour la consultation dans le sens tant direct qu'inverse. Par exemple, une fonction ATT_lookup peut être définie, l'adresse étant ainsi consultée comme un paramètre:

- adresse IPv6Address déterminée par l'intermédiaire de la fonction ATT_lookup d'une adresse DL16Address;
- adresse DL16Address déterminée par l'intermédiaire de la fonction ATT_lookup d'une adresse IPv6Address.

Plusieurs NLE ou DLE peuvent être empaquetées dans un seul appareil physique pour prendre en charge le rattachement multiple. Bien qu'une telle opération ne soit pas spécifiée par la norme, elle n'est pas interdite.

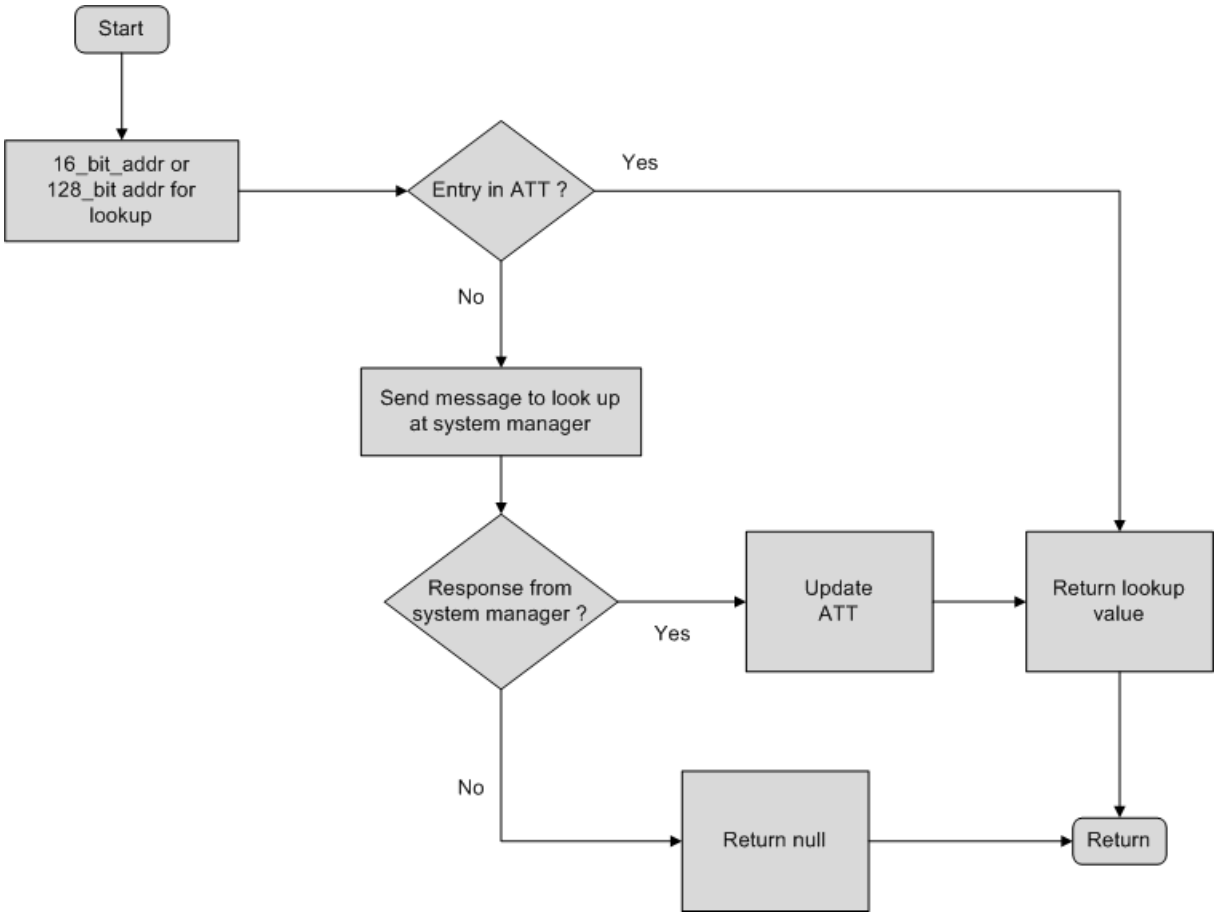
Une adresse sans entrée dans la table ATT doit être convertie avec l'aide du gestionnaire de système. Pour chaque NLE rejoignant le réseau, le gestionnaire de système doit maintenir

l'adresse IPv6Address de la NLE et l'adresse DL16Address associée ou le pseudonyme D associé pour chaque sous-réseau D dans lequel la NLE a un tel pseudonyme. Par conséquent, la table ATT locale en une NLE doit être mise à jour par l'intermédiaire du gestionnaire de système.

La table ATT est une partie intégrante du NLMO et peut être directement mise à jour par le gestionnaire de système en utilisant les méthodes de manipulation de NLMO décrites dans le Tableau 210.

Si une consultation dans la table ATT ne donne pas de résultat, la fonction de consultation le notifie au NLMO. Le NLMO délivre une primitive de lecture vers l'objet de service d'annuaire (DSO) dans le gestionnaire de système pour obtenir la conversion appropriée. La fonction de consultation retourne une valeur null (vide) si le gestionnaire de système n'a également pas de mapping pour une adresse particulière ou si le gestionnaire de système n'est pas disponible. Toute nouvelle information provenant du gestionnaire de système est stockée dans la table ATT.

Ce processus est montré à la Figure 91.



Légende

Anglais	Français
Start	Début
16_bit_addr or 128_bit_addr for lookup	16_bit_addr ou 128_bit_addr pour consultation
Entry in ATT ?	Entrée dans ATT?
No	Non
Send message to look up at system manager	Envoyer message pour consultation au gestionnaire de système
Response from system manager ?	Réponse du gestionnaire de système?

Anglais	Français
Yes	Oui
Update ATT	Mettre à jour ATT
Return lookup value	Retourner la valeur de consultation
No	Non
Return NULL	Retourner NULL
Return	Retour

Figure 91 – Processus de conversion d'adresses

10.2.4 En-têtes d'unités de données de protocoles de réseau

Trois formats sont utilisés pour les en-têtes de NPDU. La valeur du premier octet de l'en-tête fournit le moyen de distinguer entre ces formats:

- en-tête de base: ce format est destiné aux NPDU traversant un seul sous-réseau D et ne doit être utilisé que sur le sous-réseau D en question. Il est censé être le plus courant format en utilisation parce que son utilisation réduit au maximum le surdébit associé à l'émission d'en-têtes. L'en-tête de base est juste une abréviation pour une valeur fixe spécifique de l'en-tête compressé 6LoWPAN; cette valeur indique que les adresses de source et de destination sont élidées, au lieu d'être acheminées dans l'en-tête de DPDU, comme décrit en 10.5.2;
- en-tête activé par contrat: ce format également est utilisé seulement sur un seul sous-réseau D, lorsque l'appareil source a besoin d'inclure plus d'informations dans la NPDU, telles qu'un contractID. Ces informations complémentaires permettent à des routeurs dorsaux de sélectionner les ressources appropriées (par exemple, graphID, priorité) pour le routage de la NPDU, comme décrit en 10.5.3;
- en-tête complet: il s'agit d'un en-tête IPv6 complet, adapté à être utilisé sur la dorsale. Les NPDU contenant un en-tête de base ou l'en-tête activé par contrat doivent être allongées au format d'en-tête complet avant le routage sur la dorsale. En retour, les routeurs dorsaux convertissent les en-têtes complets en en-têtes de base ou en en-têtes activés par contrat pour l'émission sur un sous-réseau D, comme décrit en 10.5.4.

10.2.5 Fragmentation et réassemblage

Si la NPDU entière est plus petite que la taille de DSDU maximale, la NPDU ne doit pas être fragmentée et l'en-tête de réseau ne doit pas contenir un en-tête de fragmentation. Si la NPDU excède la taille maximale de DSDU, la NPDU doit être fragmentée en NPDU fragmentées qui ne dépassent pas la taille maximale de DSDU dans le sous-réseau D. La fragmentation doit être accomplie par la NLE au point d'entrée d'un sous-réseau D. Le réassemblage doit être accompli par la NLE au point de sortie d'un sous-réseau D.

NOTE L'émission par une TLE dans un appareil connecté à un sous-réseau D constitue “un point d'entrée” du sous-réseau D. De même, la livraison à une TLE dans un appareil connecté à un sous-réseau D constitue “un point de sortie” du sous-réseau D.

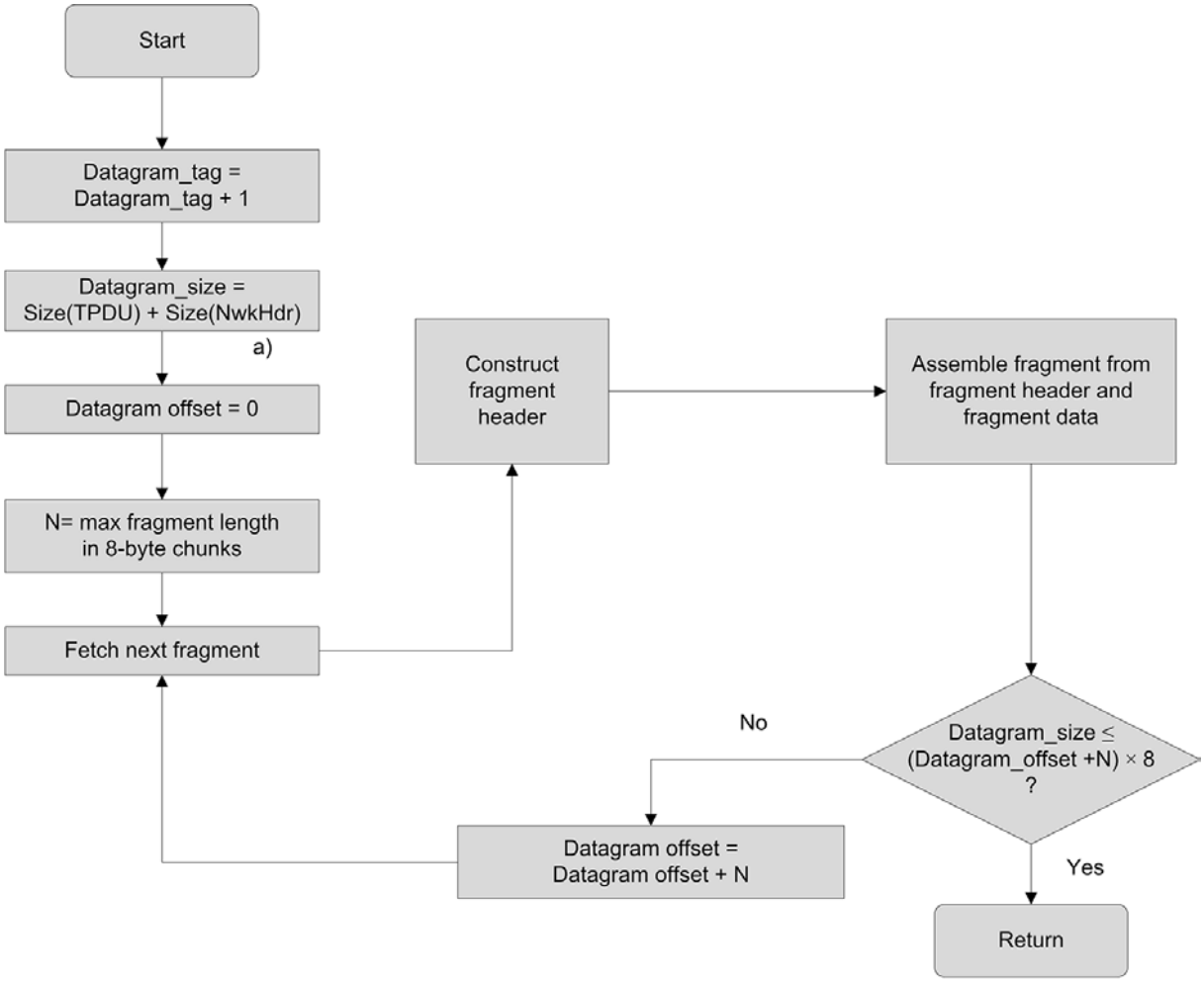
Le premier fragment doit contenir le premier en-tête de fragment tel que défini dans le Tableau 219. Le deuxième segment et les suivants (jusqu'au dernier segment compris) doivent contenir un en-tête de fragmentation, tel que définis dans le Tableau 220. Le décalage de ce fragment, appelé décalage de datagramme, doit être exprimé en unités de huit octets, de sorte que chaque fragment autre que le dernier soit constitué d'un multiple de huit octets.

Le champ Datagram_size doit être présent dans chaque fragment, pour simplifier les tâches de réassemblage lorsque les fragments arrivent dans le désordre à leur NLE de réassemblage. L'inclusion du champ Datagram_size dans chaque fragment permet au récepteur d'allouer la quantité appropriée de l'espace tampon lorsque le premier fragment est différé.

Tous les fragments (le premier et les fragments suivants) doivent avoir un champ Datagram_tag dans leur en-tête. La valeur de ce champ doit être assignée par l'appareil accomplissant la fragmentation et doit être la même pour tous les fragments de la NPDU, afin que l'appareil de réassemblage puisse reconnaître que les fragments appartiennent à la même NPDU. Dans la mesure du possible, la NLE accomplissant la fragmentation doit assigner une valeur de Datagram_tag différente à chaque NPDU distincte qu'elle fragmente. Pour ce faire, chaque NLE doit avoir un compteur qui est initialisé à une valeur aléatoire uniforme et est incrémenté pour chaque NPDU qui subit la fragmentation; la valeur de ce compteur doit être placée dans le champ Datagram_tag de chaque fragment de la NPDU.

Dans le cas extrêmement rare où deux NPDU allant de la même source vers la même destination seraient fragmentées par des routeurs intermédiaires différents qui prendraient par coïncidence le même Datagram_tag exactement, l'appareil accomplissant le réassemblage peut ne pas être capable de désambiguïser des fragments. Dans ce cas, le GraphID peut être utilisé pour désambiguïser de façon plus poussée; cependant, cela n'est pas spécifié comme étant obligatoire dans la présente norme. Les TPDU réassemblées par erreur à partir de plusieurs sources seront rejetées en raison d'erreurs de somme de contrôle puis retransmises. Les routeurs intermédiaires qui fragmentent des NPDU peuvent aussi coordonner leurs diagrammes d'états de fragmentation afin d'éviter les scénarios dans lesquels l'appareil accomplissant le réassemblage pourrait ne pas être capable de désambiguïser des fragments.

La Figure 92 montre le processus de fragmentation.



a) This is the size of the NwkHdr excluding any contained fragmentation subheader.

Légende

Anglais	Français
Start	Début
Datagram offset = 0	Décalage de datagramme = 0
N = max fragment length in 8-byte chunks	N = longueur max de fragment en morceaux de 8 octets
Fetch next fragment	Rechercher le prochain segment
Construct fragment header	Construire l'en-tête de fragment
Assemble fragment from fragment header and fragment data	Assembler le fragment à partir de l'en-tête de fragment et les données de fragment
No	Non
Datagram offset = Datagram offset + N	Décalage datagramme = Décalage datagramme + N
Yes	Oui
Return	Retour
a) This is the size of the NwkHdr excluding any contained fragmentation subheader.	a) Il s'agit de la taille de l'en-tête de réseau en excluant tout sous-en-tête de fragmentation contenu

Figure 92 – Processus de fragmentation

Pour identifier tous les fragments qui appartiennent à la même NPDU, la NLE accomplissant le réassemblage doit utiliser:

- l'adresse IPv6Address source;
- l'adresse IPv6Address de destination;
- le datagram_tag et
- le datagram_size.

Autrement, la NLE doit commencer à reconstruire la NPDU non fragmentée initiale, dont la taille est Datagram_size, en utilisant le champ Datagram_offset pour déterminer l'emplacement relatif des différents fragments au sein de la NPDU non fragmentée initiale.

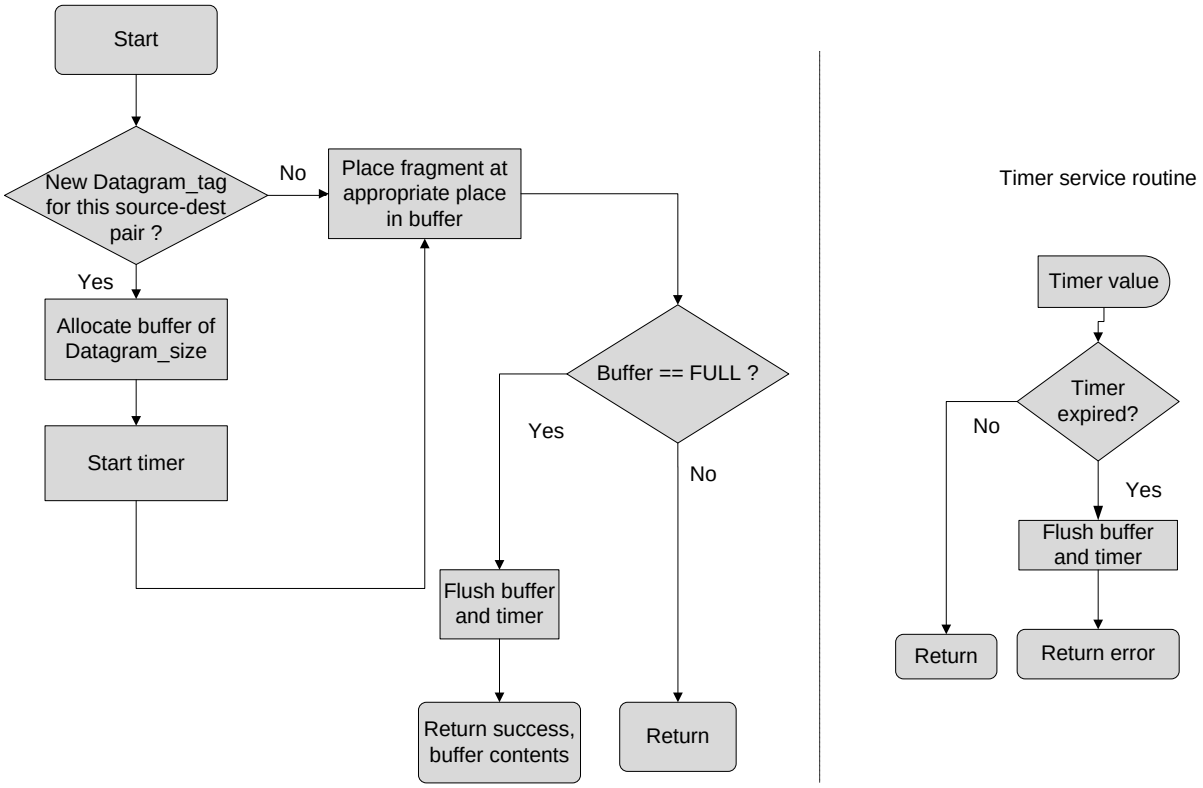
Lorsqu'une NLE reçoit d'abord un fragment avec un Datagram_tag donné qui nécessite une reconstruction, elle démarre un temporisateur de réassemblage. Si ce temporisateur expire avant que la NPDU entière n'ait été rassemblée, les fragments reçus doivent être rejetés. La temporisation du réassemblage doit être mise à une valeur définie dans NLMO.Frag_Reassembly_Timeout (identificateur d'attribut 11 dans le Tableau 206). Si un fragment qui recouvre partiellement un autre fragment est reçu, et il diffère par la taille ou par le Datagram_offset du fragment recouvert, le(s) fragments déjà accumulé(s) dans le tampon de réassemblage doit/doivent être rejeté(s).

Le texte précédant la Figure 92 fournit un exemple de la façon dont une telle fragmentation inconsistante peut se produire.

Un nouveau réassemblage débute avec un fragment contenant une étiquette pour laquelle aucun fragment n'est en cours. Il peut en résulter que des tampons sont alloués quand certains fragments arrivent après la temporisation du processus de réassemblage qui avait été précédemment initié pour la même étiquette (par essence, en tentant de réassembler la NPDU une nouvelle fois ou plus tard). En général, ce réassemblage répété échouera, entraînant l'éventuelle purge de nouveaux tampons à cause de NLMO.Frag_Reassembly_Timeout.

Le processus de réassemblage est achevé lorsque la NPDU est entièrement réassemblée ou lorsque le temporisateur expire. Si la NPDU excède la taille indiquée par NLMO.Max_NSDU_size, le processus de réassemblage peut être abandonné et la NPDU peut être rejetée. L'appareil peut envoyer une alerte PDU abandonnée/erreur de PDU avec la valeur 7 indiquant qu'il manque de mémoire. Les alertes PDU abandonnée/erreur de PDU sont montrées dans le Tableau 211.

Le processus de réassemblage de NPDU est montré à la Figure 93.



Légende

Anglais	Français
Start	Début
New Datagram_tag for this source-dest pair ?	Nouveau Datagram_tag pour cette paire source-destination ?
No	Non
Place fragment at appropriate place in buffer	Placer fragment à l'endroit approprié dans tampon
Yes	Oui
Allocate buffer of Datagram_size	Allouer un tampon de Datagram_size
Start timer	Démarrer temporisateur
Buffer == FULL ?	Tampon == PLEIN?
Flush buffer and timer	Purger tampon et temporisateur
Return success, buffer contents	Retourner succès, contenu tampon
Return	Retour
Timer service routine	Routine de service du temporisateur
Timer value	Valeur de temporisateur
Timer expired?	Temporisateur expiré?
Flush buffer and timer	Purger tampon et temporisateur
Return error	Erreur de retour

Figure 93 – Processus de réassemblage

10.2.6 Routage

10.2.6.1 Généralités

Le routage dans un réseau conforme à la présente norme se produit à deux niveaux:

- un niveau comporte les points d'extrémité et les appareils dorsaux, s'il y en a; la NL est responsable du routage des PDU à ce niveau. Ce niveau ne traite pas le routage sur les liaisons de DL; la traversée d'un sous-réseau D apparaît comme un seul saut à une NLE;
- le deuxième niveau de routage se situe au sein d'un sous-réseau D. Ce niveau est de la responsabilité de la DL (une mise en œuvre de maille de couche 2).

Le routage entre sous-réseaux D et réseaux dorsaux est de la responsabilité de la NL, dont les NPDU sont conformes aux normes IETF IPv6 et 6LoWPAN La présente norme spécifie des exigences minimales pour le routage, avec des services de gestion hypothétiques pour ajouter, supprimer et maintenir des chemins.

10.2.6.2 Tables de routage

La NLE dans des appareils conformes à la présente norme doit maintenir une table de routage (RT) pour garder la trace du prochain saut pour une destination donnée. Cette table doit être maintenue en utilisant des adresses IPv6Addresses, car de telles adresses sont uniques à travers un réseau entier conforme à la présente norme (y compris tous les sous-réseaux D). Un exemple d'une table de routage est fourni dans le Tableau 202. La table de routage peut être mise à jour à l'appareil de source chaque fois qu'une session de communication est établie avec un appareil de destination.

Tableau 202 – Exemple de table de routage

DestinationAddress	NextHop	NWK_Hop_Limit	OutgoingInterface [a]
N1	BBR1	2	Dorsale
N2	BBR1	2	Dorsale
GW	GW	2	Dorsale
N3	N3	1	sous-réseau D
N4	N4	1	sous-réseau D
N5	N5	1	sous-réseau D
…	…	…	…

[a] Ce champ est mis au sous-réseau D pour toutes les destinations dans les routeurs et les appareils E/S.

NOTE Dans la présente norme, la table de routage et tous les objets de gestion de NL sont spécifiés pour prendre en charge un seul sous-réseau D actif à la fois. Toutes les adresses DL16Addresses sont uniques dans la portée du seul sous-réseau D en question. Cela n'est pas censé empêcher un appareil de participer simultanément à plusieurs sous-réseaux D. Plusieurs sous-réseaux D sont représentés par plusieurs NLE.

Les DLE qui ne sont pas à capacité de dorsale acheminent seulement des DPDU au sein du sous-réseau D. Le routage au sein du sous-réseau D est de la responsabilité de la DL (une mise en œuvre de maille de couche 2). Par conséquent, les DLE qui fonctionnent dans le sous-réseau D, mais qui ne sont pas à capacité de dorsale, peuvent maintenir une table de routage, mais elles ne sont pas tenues de le faire. Cela est également reflété dans le Tableau B.18 qui propose de façon normative les tailles minimales des tables de routage qui ont besoin d'être prises en charge par les appareils qui répondent à divers profils de rôles. Les tables de routage de NL fournissent l'indépendance de couche et permettent des mises en œuvre potentielles de route-over, où le routage dans le sous-réseau D est réalisé par l'intermédiaire du routage de NL.

La table de routage doit être également utilisée par les routeurs dorsaux pour décider d'acheminer une PDU sur une dorsale ou sur le sous-réseau D de la présente norme. Le champ OutgoingInterface indique si la PDU doit être envoyée sur la dorsale ou sur le sous-réseau D.

NextHop indique le prochain appareil dont la NLE doit traiter la NPDU destinée à la DestinationAddress. N'importe quel appareil joignable par l'intermédiaire de la maille de DL a NextHop égal à l'adresse de destination et le champ NWK_Hop_Limit mis à 1. De la perspective de la NLE, tout appareil qui est joignable par l'intermédiaire de la maille de DL est à un seul saut de réseau de distance.

10.2.6.3 Traitement d'une unité de données de service réseau reçue en provenance d'une TLE

Lorsqu'une NSDU est passée à une NLE par une TLE, la NLE détermine la destination finale pour la NSDU en question en se basant sur le ContractID. La table de contrat (voir Tableau 207) est utilisée pour obtenir l'adresse de destination. Les appareils avec une dorsale et une interface de DL conforme à la présente norme doivent rechercher l'adresse de destination dans la table de routage pour déterminer quelle interface réseau utiliser. Toutes les DLE autres que dorsales doivent toujours utiliser leur interface de DL.

La NLE doit utiliser la ContractTable pour obtenir la priorité à deux bits pour le contractID dans la N-Data.request. Cette priorité de contrat doit être combinée avec les deux bits de la priorité de message (également passée dans la N-Data.request) pour obtenir une priorité de NPDU à 4 bits qui est passée à la DLE; les deux bits de poids fort doivent être la priorité de contrat, et les deux bits de poids faible doivent être la priorité de message. Le champ Discard Eligible (DE) issu de la N-Data.request est également passé à la DLE. Si l'OutgoingInterface pour l'adresse de destination est la dorsale, alors la priorité de 4 bits et les bits éligibles de DE doivent être inclus dans le champ TrafficClass de l'en-tête IPv6.

La NLE doit utiliser la ContractTable pour vérifier si le ContractID a besoin d'être inclus dans la NPDU. Le fait d'inclure le ContractID dans la NPDU permet aux routeurs dorsaux intermédiaires de faire des choix appropriés de routage (niveau de service, graphID, etc.) sur la dorsale ou sur un sous-réseau D différent. Pour le routage sur l'interface de DL, si le ContractID n'a pas besoin d'être inclus, il convient alors qu'un en-tête de base de NPDU soit construit; autrement, il convient de construire un en-tête de NPDU activé par contrat.

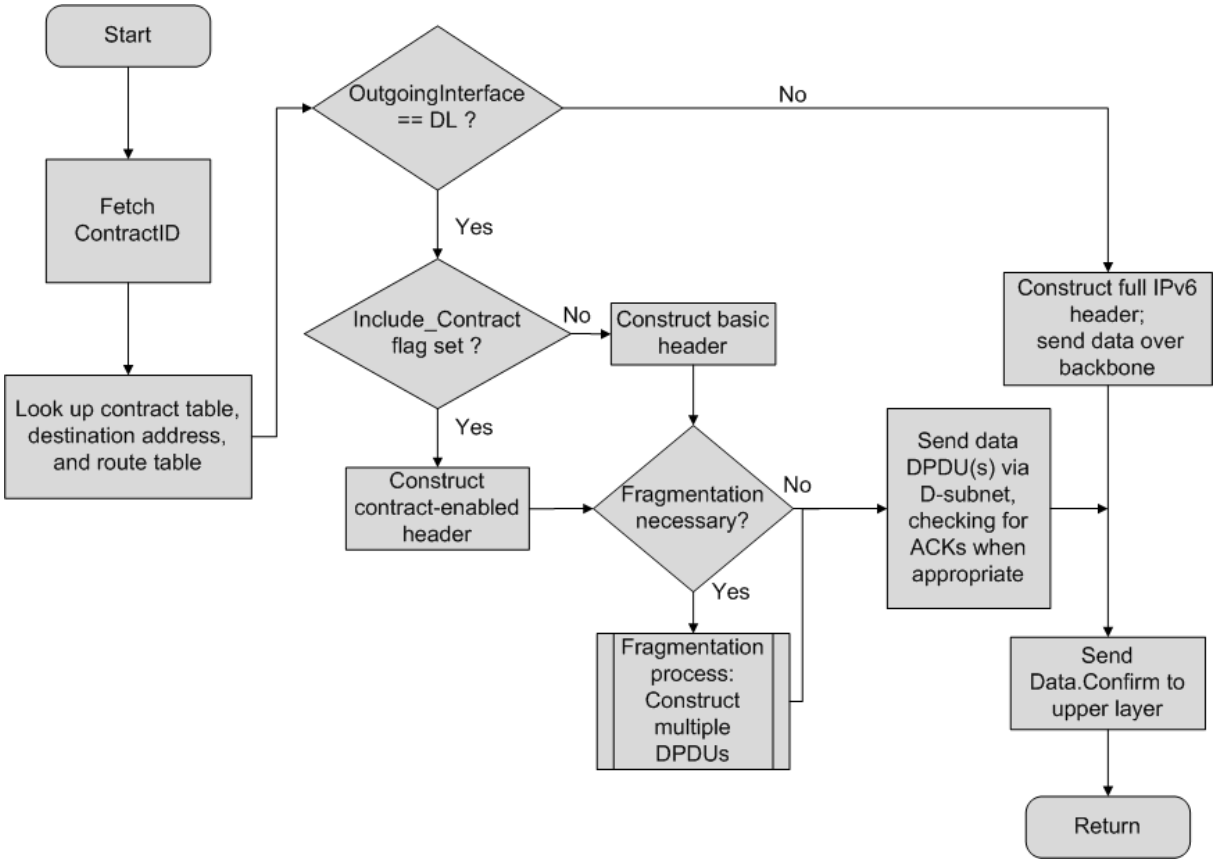
La NLE doit aussi déterminer si, oui ou non, la fragmentation est nécessaire pour la NPDU et doit accomplir le processus de fragmentation s'il y a lieu. La fragmentation doit être exigée seulement pour les NPDU acheminées sur un sous-réseau D; l'attribut dlmo.MaxDSDUSize doit indiquer la charge utile maximale qui peut être transportée sur un sous-réseau D. Si la taille de DSDU est supérieure à cette valeur, la fragmentation est alors nécessaire.

Les mécanismes de mise en cache des BBR (routeurs dorsaux intermédiaires) ainsi que les protocoles de transmission et de réassemblage inter-BBR peuvent fournir les fonctionnalités nécessaires pour permettre aux fragments de NSDU qui arrivent (du sous-réseau D) de différer le réassemblage et la transmission des BBR par l'un de ces BBR.

NOTE Une édition future de la présente norme peut spécifier un tel mécanisme et protocole inter-BBR.

A moins que le gestionnaire de système de configuration ne sache que les BBR sélectionnés ont la capacité nécessaire, les NPDU exigeant une fragmentation ne doivent pas utiliser les chemins du sous-réseau D qui se terminent par plusieurs BBR, car les NPDU non initiales résultant d'une fragmentation 6LoWPAN ne transportent pas suffisamment d'informations pour qu'elles soient acheminées directement vers leur destination finale prévue sur le sous-réseau dorsal avant que le réassemblage n'ait lieu.

La Figure 94 montre le traitement d'une NSDU reçue en provenance d'une TLE.



Légende

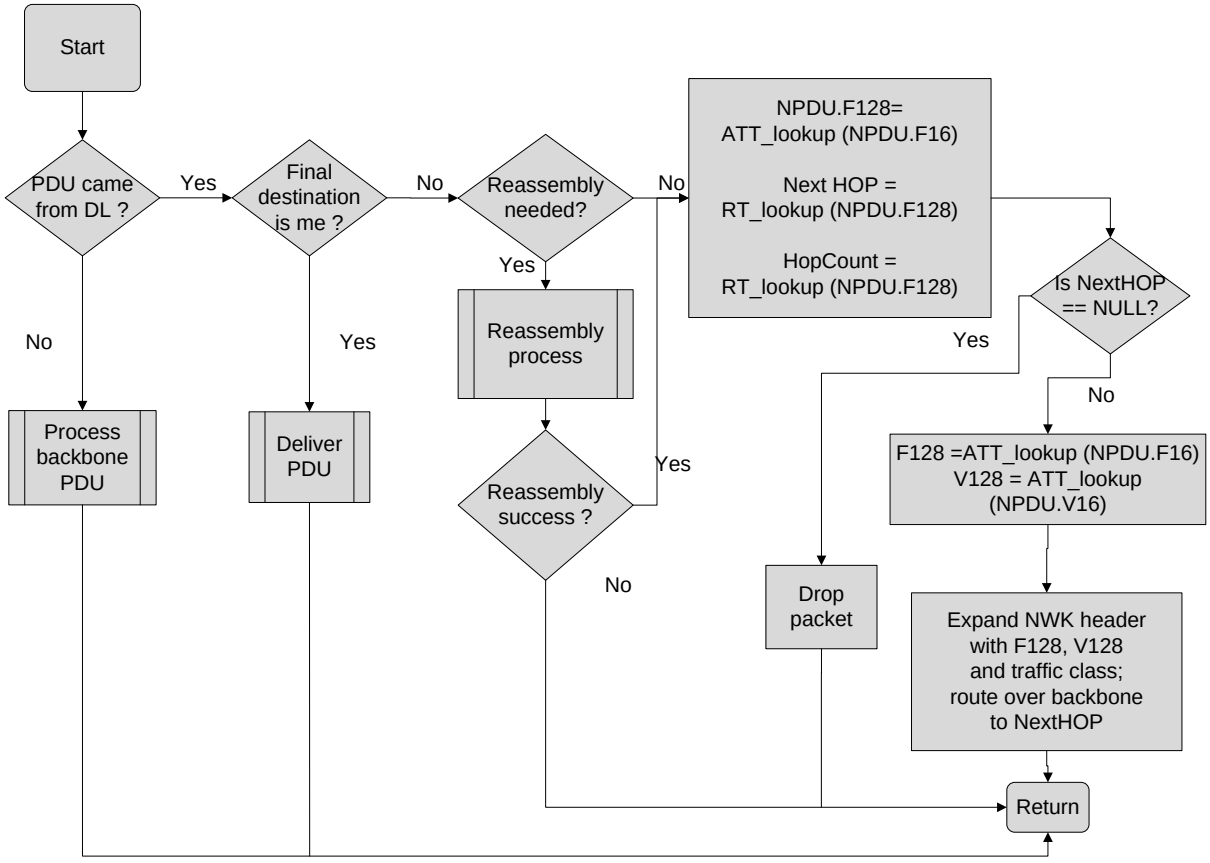
Anglais	Français
Start	Début
Fetch ContractID	Rechercher ContactID
Look up contract table, destination address and route table	Rechercher contract table.destination address et table de routage
Yes	Oui
Include_Contract flag set ?	Fanion Include_Contract mis?
Construct contract–enabled header	Construire en-tête activé par contrat
No	Non
Construct basic header	Construire en-tête de base
Fragmentation necessary?	Fragmentation nécessaire?
Fragmentation process: Construct multiple DPDUs	Processus de fragmentation: construction de plusieurs DPDU
Send data DPDU(s) via D-subnet, checking for ACKs when appropriate	Envoyer les DPDU de données via le sous-réseau D, vérification des ACK si nécessaire
Construct full IPv6 header: send data over backbone	Construire en-tête IPv6 complet; envoyer données sur dorsale
Send Data.Confirm to upper layer	Envoyer Data.Confirm à la couche supérieure
Return	Retour

Figure 94 – Traitement d'une NSDU reçue en provenance d'une TLE

10.2.6.4 Traitement d'une NPDU reçue

Une NPDU reçue (un paquet), qu'elle soit reçue en provenance de DL ou en provenance de l'interface de dorsale, doit être d'abord vérifiée pour déterminer si la destination finale est une TLE de l'appareil courant. Si oui, la NSDU qui est acheminée comme étant la charge utile de la NPDU (après une éventuelle défragmentation de NPDU exigée) doit être passée à la TLE

contiguë. Si la destination finale n'est pas l'appareil courant, l'appareil doit acheminer la NPDU de façon appropriée (par l'intermédiaire soit de la dorsale, soit de la DLE associée). Le processus global de décision est montré à la Figure 95. Tous les paquets reçus en provenance de la DLE n'auront pas une entrée d'adresse DL16Address correspondante dans la table ATT. Certains appareils fonctionnant sur la dorsale peuvent ne pas avoir une adresse DL16Address assignée, mais avoir uniquement une adresse IPv6Address. Dans ce cas, la NPDU sera livrée par la DLE locale après avoir été transmise à partir de la DLE distante expéditrice sur un chemin par défaut. Dans ce cas, l'appareil à capacité de dorsale recherchera directement le chemin associé à l'adresse IPv6Address de destination.



Légende

Anglais	Français
Start	Début
PDU came from DL ?	PDU est venu de DL?
No	Non
Process backbone PDU	Traiter PDU de dorsale
Yes	Oui
Final destination is me?	Destination finale est moi?
Deliver PDU	Livrer la PDU
Reassembly needed ?	Réassemblage nécessaire?
Reassembly process	Processus de réassemblage
Reassembly success?	Succès du réassemblage?
Is NextHOP == NULL?	Est-ce que NextHOP == NULL?
Drop packet	Abandonner le paquet
Expand NWK header with F128, V128 and traffic class; route over backbone to NextHOP	Etendre l'en-tête NWK avec F128, V128 et classe de trafic; acheminer sur dorsale vers NextHop
Return	Retour

NOTE La coordination des BBR pour le réassemblage des paquets de NPDU fragmentées par 6LoWPAN pour l'émission à travers la capacité très limitée de charge utile du sous-réseau D est spécifique au fournisseur et ne relève pas du domaine d'application de la présente norme.

Figure 95 – Traitement d'une NPDU reçue

Les services hypothétiques DD-DATA.indication et DD-DATA.request de la DLE acheminent un paramètre LastHop (LH). Lorsque ce paramètre LH est mis, il indique que la PDU est entrée dans le sous-réseau D par un routeur dorsal, et qu'elle ne doit donc pas sortir du sous-réseau D par un routeur dorsal afin d'éviter les chemins circulaires au sein de la NL. Cette restriction permet à la NLE d'élider le champ Hop Limit d'une NPDU compressée qui utilise le format d'en-tête de base tout en empêchant toujours le routage circulaire.

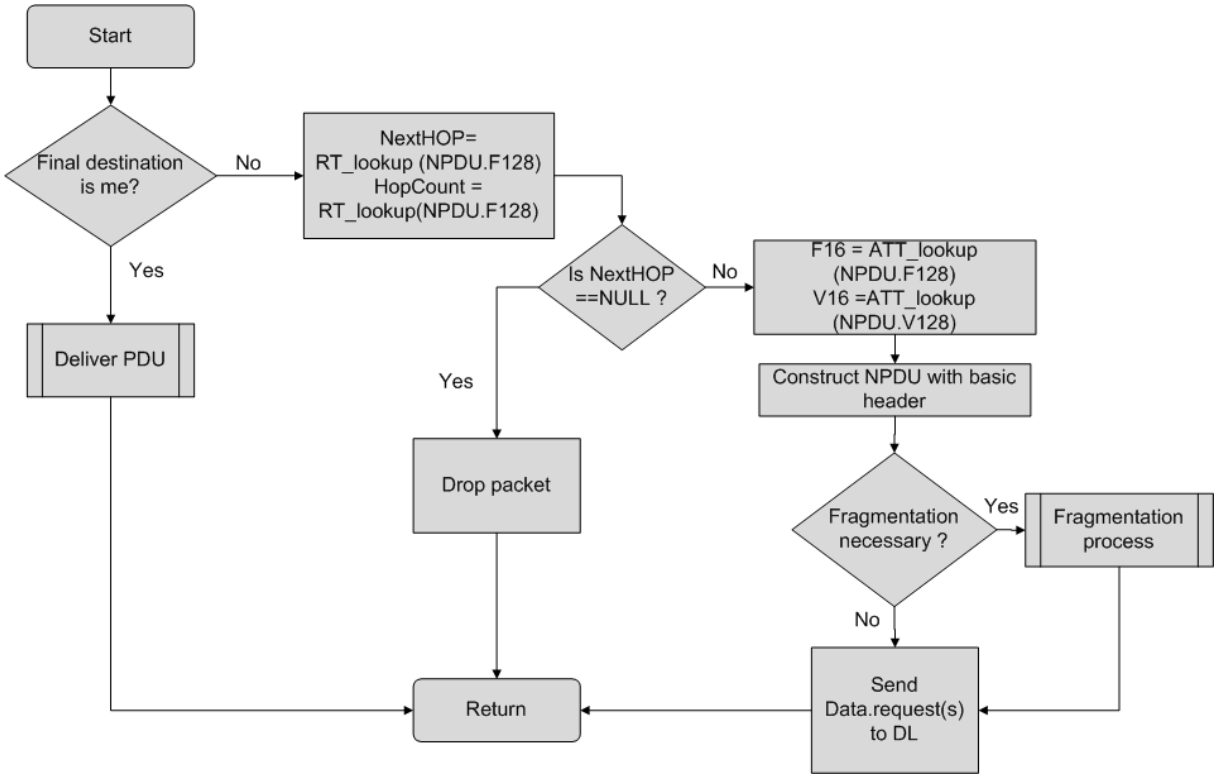
Lorsque la NPDU est reçue en provenance de la DL à un appareil autre que la destination, si le paramètre LastHop (LH) est mis dans la primitive DD-DATA.indication, la NPDU a atteint l'appareil courant en erreur et doit être rejetée. Si la NPDU est reçue en provenance du sous-réseau D et n'est pas rejetée (voir Figure 95), le routeur intermédiaire doit d'abord étendre complètement l'en-tête du réseau de la NPDU. Comme partie intégrante de cette extension, la valeur de la notification d'encombrement explicite (ECN) fournie par la primitive DD-DATA.indication doit être incluse dans le champ approprié de l'en-tête étendu.

Après toute expansion d'en-tête, la DLE réceptrice doit vérifier pour voir si, oui ou non, le réassemblage (en raison d'une fragmentation antérieure) est nécessaire pour cette NPDU. Une fois que l'éventuel réassemblage nécessaire s'achève, la NPDU doit être préparée pour un routage sur la dorsale. Les adresses DL16Address d'origine (tout premier V) et de destination (destination finale F) dans la primitive DD-DATA.indication doivent être converties en adresses IPv6Addresses. Ensuite, la table de routage doit être utilisée pour déterminer le prochain saut de NL pour atteindre la destination finale. La NPDU doit être présentée à la NLE de l'interface de la dorsale pour le routage sur le réseau dorsal. La présente norme ne spécifie pas comment la dorsale traite et achemine la NPDU. La dorsale a la responsabilité de livrer la NPDU à la NLE du NextHop.

La présente norme utilise toujours ECN. Lorsque la notification d'encombrement est transportée dans un en-tête de DPDU, si les bits ECN sont différents de zéro dans l'en-tête de la NPDU, ils doivent être mis à zéro dans l'en-tête en question pour indiquer que la notification est transportée dans l'en-tête de la DPDU. Une NLE de routeur dorsal qui reçoit une NPDU potentiellement réassemblée provenant de sa DLE associée doit utiliser les informations ECN transportées dans l'en-tête de DPDU reçu pour remplir les bits ECN dans l'en-tête de NPDU étendu. Les NPDU provenant d'appareils de dorsale doivent avoir les bits ECN mis pour indiquer que la notification d'encombrement explicite est utilisée.

Si une NPDU est reçue en provenance de la dorsale, elle aura un en-tête étendu, et les adresses de destination finale et les toutes premières adresses (d'initiateur) seront déjà exprimées sous forme d'adresses IPv6Addresses. Si la NPDU a besoin d'être acheminée sur un sous-réseau D, les adresses DL16Addresses dans le sous-réseau D en question de la toute première DLE (initiatrice) et de la DLE de destination finale doivent être obtenues à partir de la table ATT et passées à la DLE dans la primitive DD-DATA.request. La NLE doit vérifier si le ContractID et la priorité sont respectivement inclus dans les champs FlowLabel et TrafficClass de l'en-tête étendu de la NPDU. Si tel est le cas, le ContractID et la priorité doivent également être passés à la DLE pour permettre la sélection des mécanismes appropriés de routage de DL (GraphID, etc.).

La présence ou l'absence d'encombrement est déterminée à partir du champ ECN de la NPDU reçue en provenance de la dorsale, qui est passée à la DLE locale dans une primitive DD-DATA.request. Lors du passage d'une NPDU avec un en-tête de base à une DLE locale, le paramètre LastHop (LH) doit être mis dans la primitive DD-DATA.request pour indiquer que la NPDU est entrée dans un sous-réseau D dont elle ne peut pas sortir. Si la taille de NPDU excède dlmo.MaxDsduSize pour ce sous-réseau D, la NPDU doit être fragmentée avant le transport sous la forme des DSDU. Ce processus est décrit sous la forme d'un organigramme à la Figure 96.

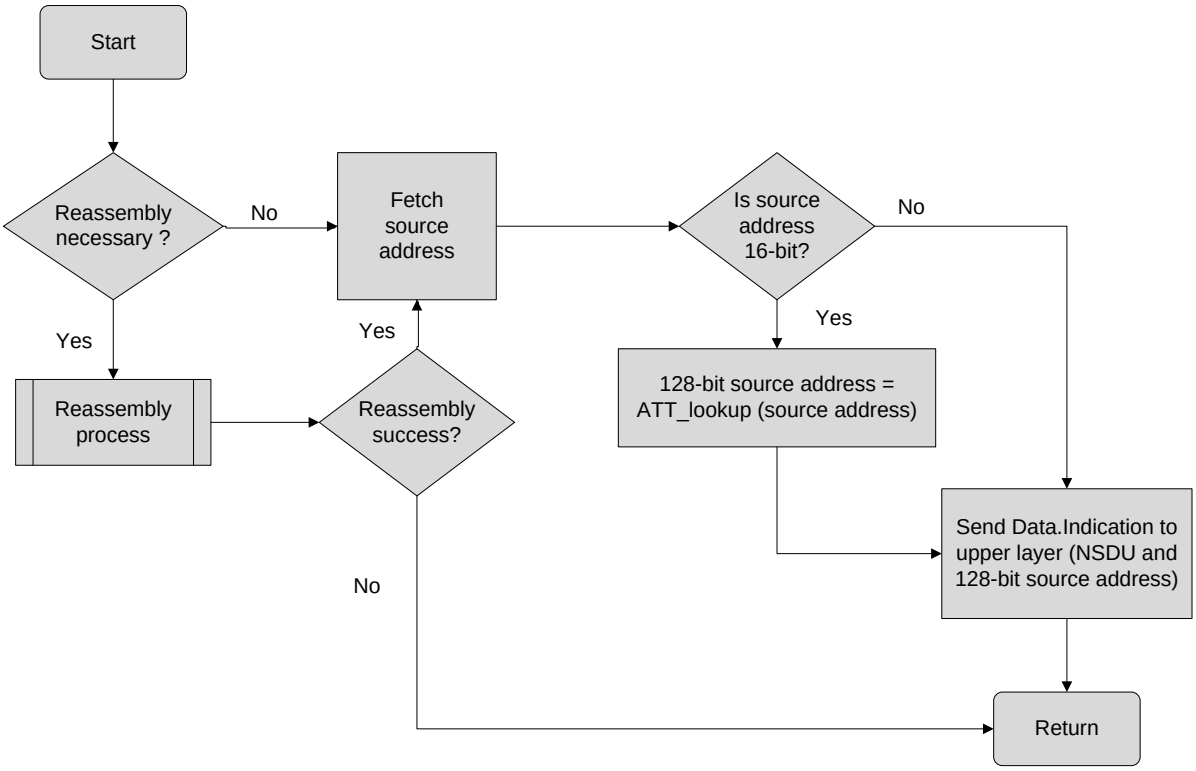


Légende

Anglais	Français
Start	Début
Final destination is me?	Destination finale est moi?
Yes	Oui
Deliver PDU	Livrer la PDU
No	Non
Is NextHOP ==NULL?	Est-ce que NextHOP ==NULL?
Drop packet	Abandonner paquet
Return	Retour
Construct NPDU with basic header	Construire NPDU avec en-tête de base
Fragmentation necessary?	Fragmentation nécessaire?
Fragmentation process	Processus de fragmentation
Send Data.request(s) to DL	Envoyer Data.request(s) à la DL

Figure 96 – Traitement d'une NPDU reçue par une NLE en provenance de la dorsale

Si la NLE réceptrice est la destination finale prévue, la NLE en question doit traiter la NPDU et doit passer à la TLE locale associée la NSDU acheminée, accompagné d'une indication disant si, oui ou non, un encombrement a eu lieu, tel qu'acheminé par les bits ECN de la NPDU. La NLE doit d'abord vérifier si elle a reçu un fragment; si tel est le cas, elle doit accomplir le processus de réassemblage (voir Figure 93). L'adresse IPv6Address source de la NPDU doit être convertie en une adresse IPv6Address, s'il y a lieu, et la NSDU doit être passée à la TLE associée. La Figure 97 décrit l'organigramme pour ce traitement.



Légende

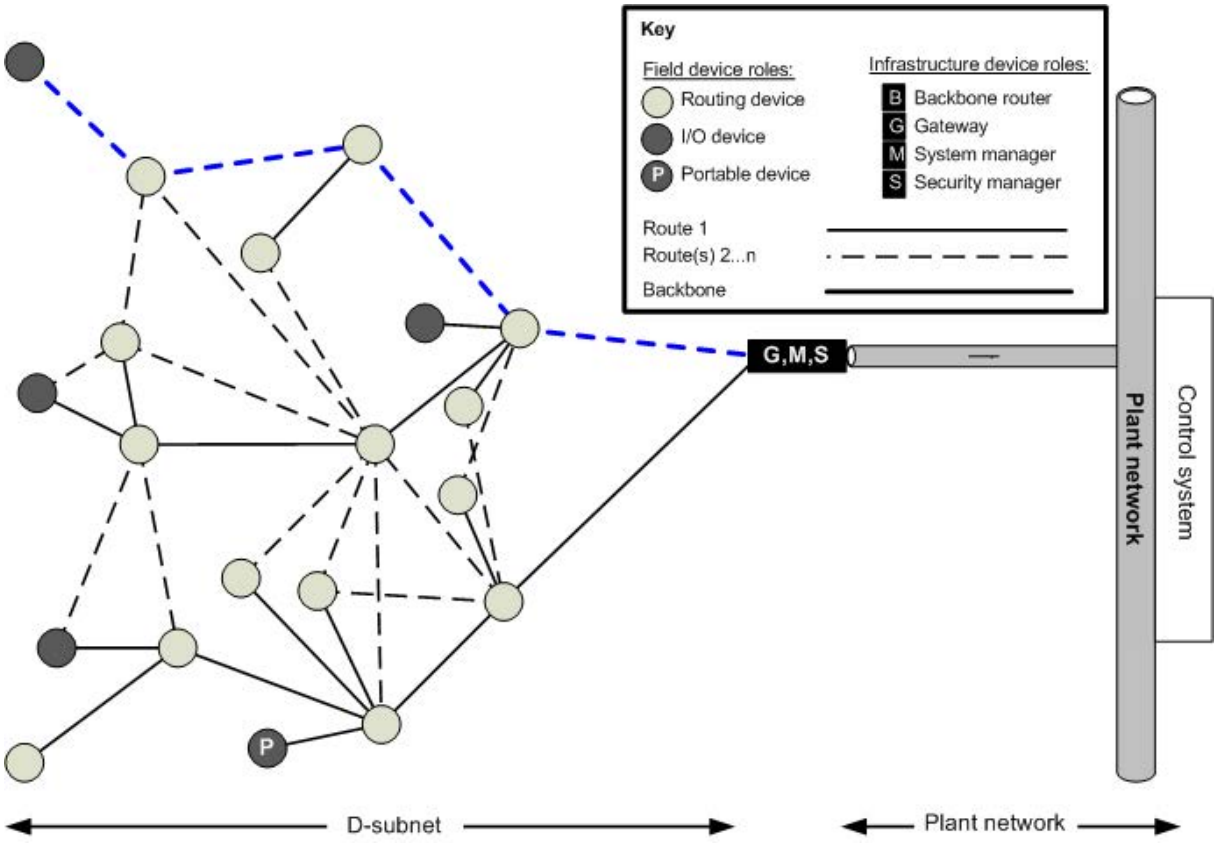
Anglais	Français
Start	Début
Reassembly necessary ?	Réassemblage nécessaire?
Yes	Oui
Reassembly process	Processus de réassemblage
No	Non
Fetch source address	Rechercher adresse source
Reassembly success?	Succès du réassemblage?
Is source address 16-bits?	Est-ce que l'adresse source est de 16 bits?
128-bit source address = ATT_lookup (source address)	Adresse source de 128 bits = ATT_lookup (adresse source)
Send Data.Indication to upper layer (NSDU and 128-bit source address)	Envoyer Data.Indication à la couche supérieure (NSDU et adresse source de 128 bits)
Return	Retour

Figure 97 – Livraison d'une NPDU reçue à sa NLE de destination finale

10.2.7 Exemples de routage

10.2.7.1 Routage à partir d'un appareil de terrain directement jusqu'à une passerelle de terrain connectée

La Figure 98 montre le routage allant d'appareil de terrain vers une passerelle sans routage dorsal.

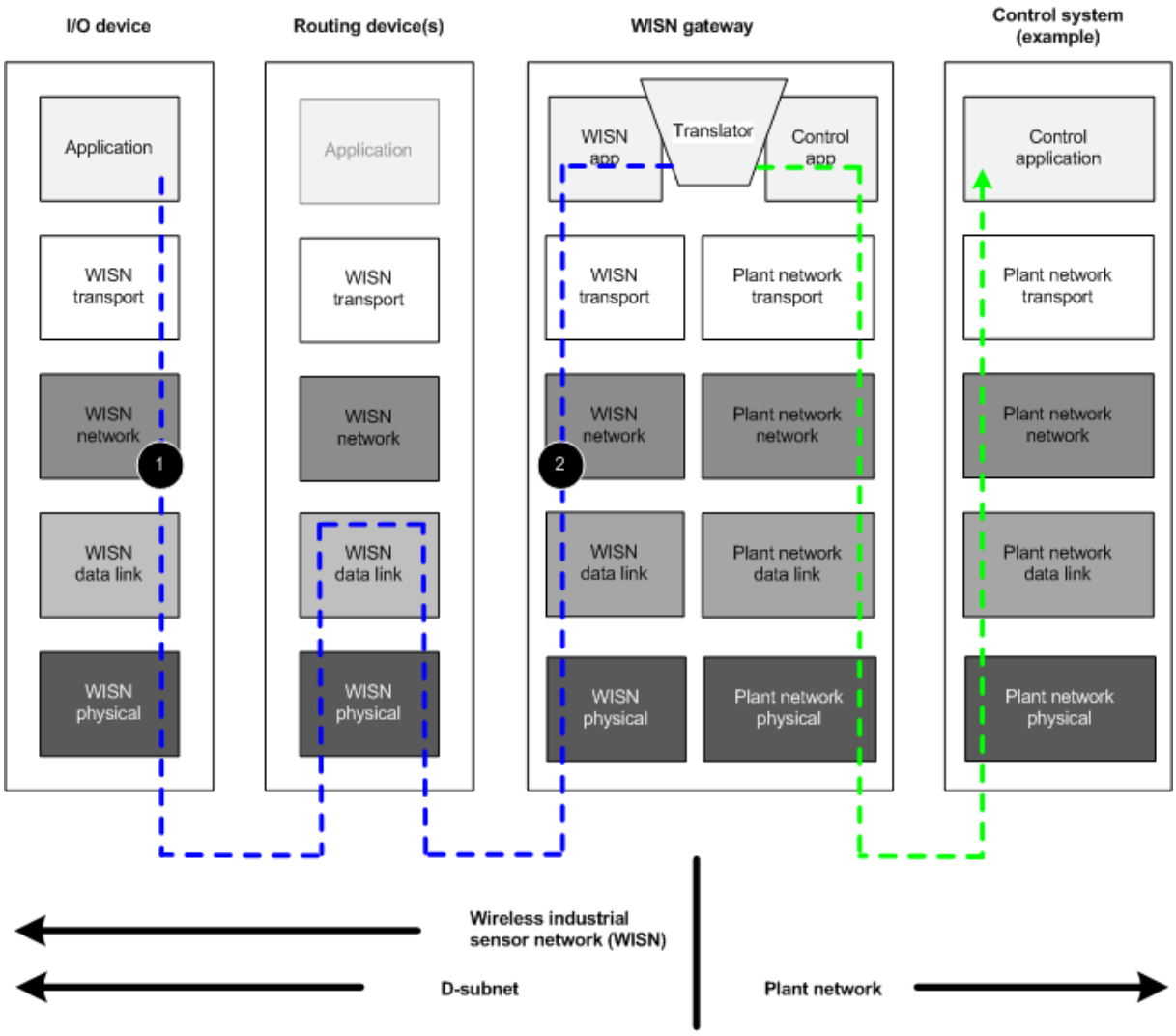


Légende

Anglais	Français
Key	Légende
Field device roles	Rôles d'appareil de terrain
Routing device	Appareil de routage
I/O device	Appareil E/S
Portable device	Appareil portatif
Infrastructure device roles	Rôles d'appareil d'infrastructure
B Backbone router	B Routeur dorsal
G Gateway	G Passerelle
M System manager	M Gestionnaire de système
S Security manager	S Gestionnaire de sécurité
Route 1	Chemin 1
Route(s) 2...n	Chemin(s) 2...n
Backbone	Dorsale
Control system	Système de commande
Plant network	Réseau d'installation
D-subnet	Sous-réseau D
Backbone network	Réseau dorsal

Figure 98 – Routage allant d'un appareil de terrain directement jusqu'à une passerelle connectée de champ sans routage dorsal

La Figure 99 montre le chemin de routage à travers les suites de protocoles de communication à mesure que la PDU se déplace d'un appareil E/S vers la passerelle. Il est supposé que la NPDU n'a besoin d'aucune fragmentation.



Légende

Anglais	Français
I/O device	Appareil E/S
Application	Application
WISN transport	Transport WISN
WISN network	Réseau WISN
WISN data link	Liaison de données WISN
WISN physical	Physique de WISN
Routing device(s)	Appareil(s) de routage
WISN gateway	Passerelles WISN
WISN app	App WISN
Translator	Convertisseur
Control app	App de commande
Plant network transport	Transport de réseau d'installation
Plant network network	Réseau de réseau d'installation
Plant network data link	Liaison de données de réseau d'installation
Plant network physical	Physique de réseau d'installation
Control system (example)	Système de commande (exemple)
Control application	Application de commande
Wireless industrial sensor network (WISN)	Réseau de capteurs industriels sans fil (WISN)
D-subnet	Sous-réseau D
Plant network	Réseau d'installation

Figure 99 – Diagramme de suites de protocoles pour le routage allant d'un appareil de terrain directement vers une passerelle connectée de terrain sans routage dorsal

A la Figure 99, la passerelle est montrée comme ayant un moyen de terrain; aucun réseau dorsal n'est donc impliqué dans cet exemple. Les pointillés bleus (à gauche) représentent le flux de messagerie sur le réseau sur l'interface inférieure (partie du terrain par exemple) de la passerelle, alors que les pointillés verts (à droite) représentent le flux de messagerie sur le réseau sur l'interface supérieure (partie de la gestion opérationnelle par exemple) de la passerelle.

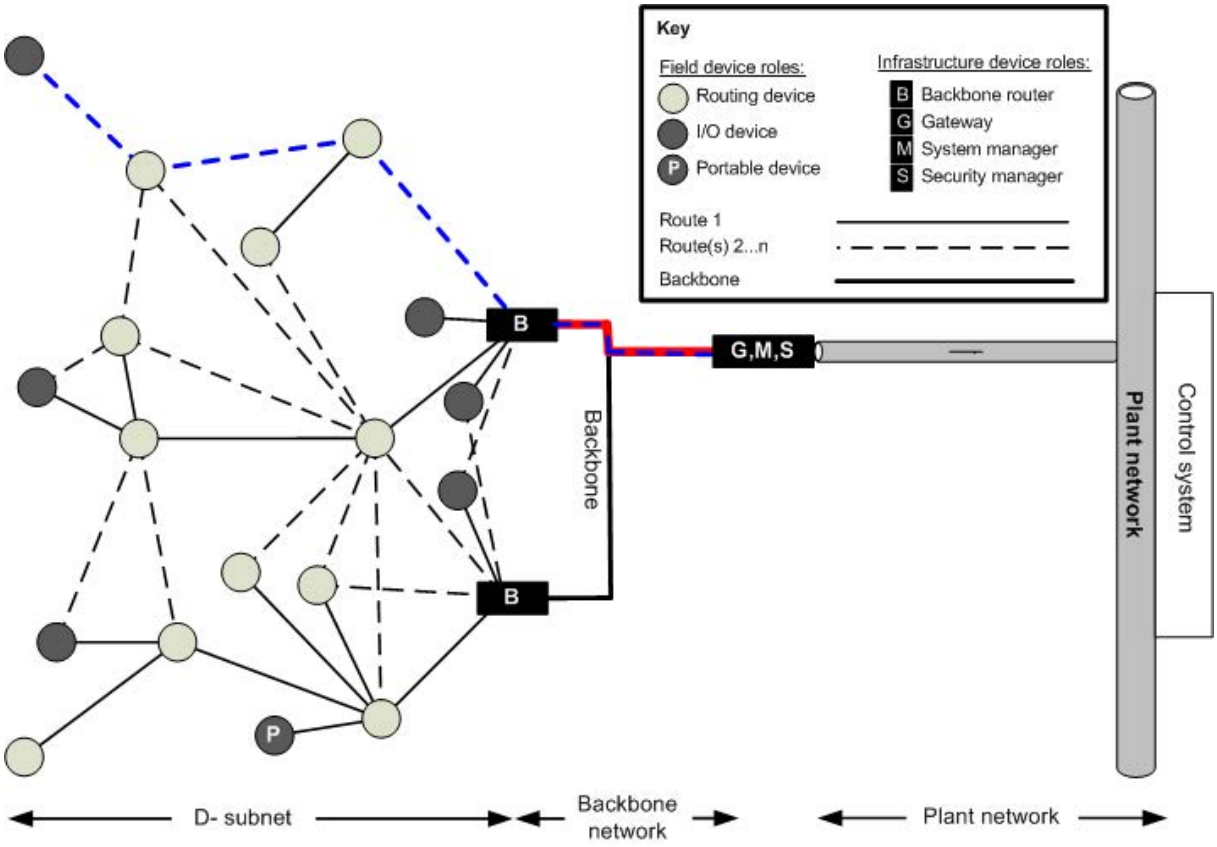
Les opérations des NLE au niveau des appareils que la NPDU traverse (numérotés dans l'ordre) sont comme suit:

a) la NLE dans l'appareil de terrain d'origine utilise un en-tête de base de réseau; l'adresse DL16Address de la passerelle et l'adresse DL16Address de l'appareil lui-même sont obtenues à partir de la table ATT et sont passées à la DLE sous forme d'une DSDU pour acheminement vers la passerelle;

b) la NLE dans la passerelle reçoit la NPDU, vérifie que la NPDU est destinée à la passerelle, convertit l'adresse DL16Address de l'appareil d'origine (fournie par la primitive DD-DATA.indication) en adresse IPv6Address, puis passe la NSDU à la TLE.

10.2.7.2 Routage allant d'un appareil de terrain vers une passerelle en passant par un routeur dorsal

La Figure 100 montre le routage d'une PDU allant d'un appareil de terrain vers une passerelle en passant par un routeur dorsal.



Légende

Anglais	Français
Key	Légende
Field device roles	Rôles d'appareil de terrain

Anglais	Français
Routing device	Appareil de routage
I/O device	Appareil E/S
Portable device	Appareil portatif
Infrastructure device roles	Rôles d'appareil d'infrastructure
B Backbone router	B Routeur dorsal
G Gateway	G Passerelle
M System manager	M Gestionnaire de système
S Security manager	S Gestionnaire de sécurité
Route 1	Chemin 1
Route(s) 2…n	Chemin(s) 2..n
Backbone	Dorsale
Control system	Système de commande
Plant network	Réseau d'installation
D-subnet	Sous-réseau D
Backbone network	Réseau dorsal

Figure 100 – Routage d'une NPDU allant d'un appareil de terrain vers une passerelle en passant par un routeur dorsal

La Figure 101 montre le flux d'une NPDU allant d'un appareil de terrain vers une passerelle résidant sur le réseau dorsal.

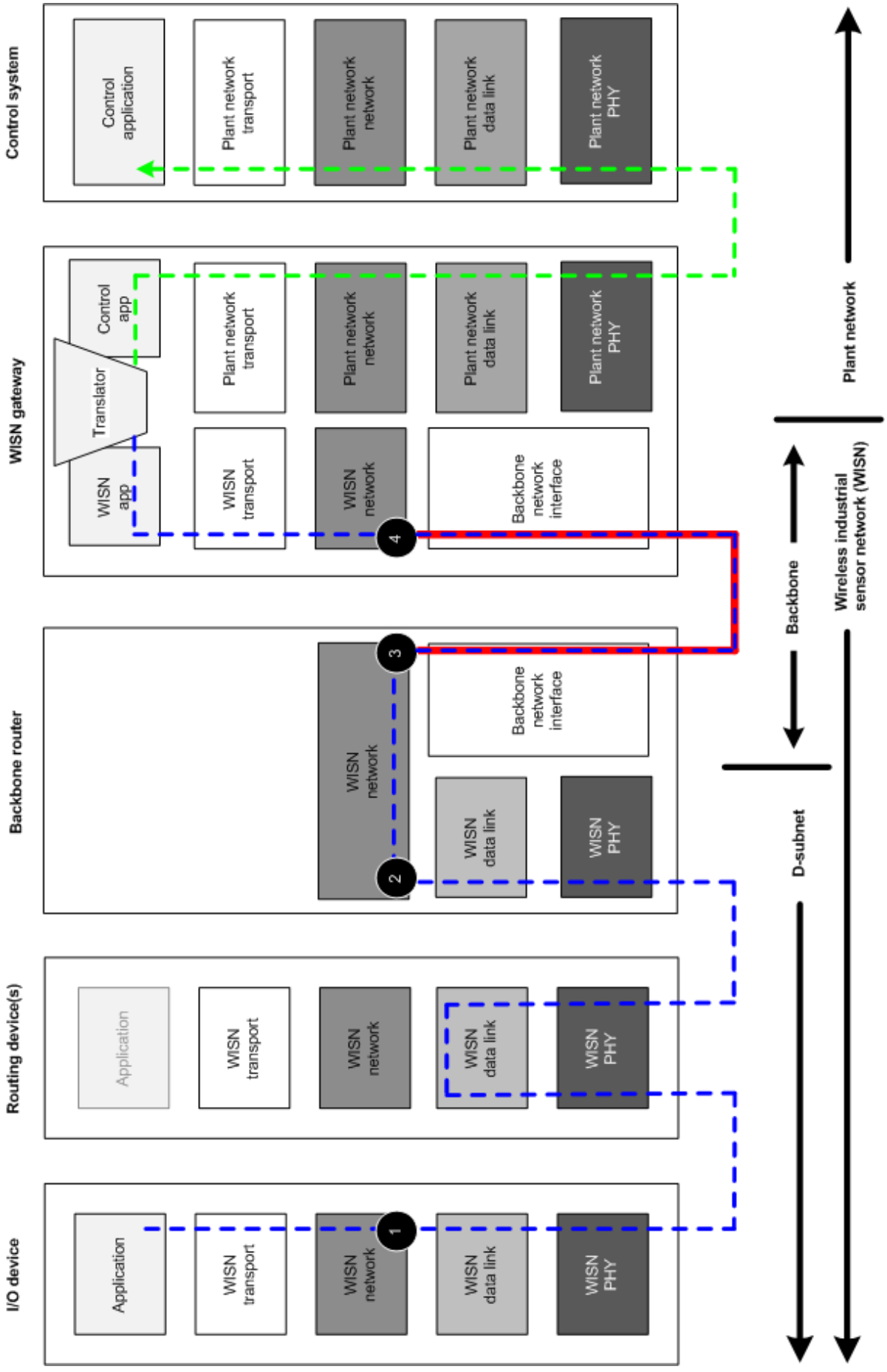
La NPDU est d'abord acheminée vers un routeur dorsal sur le sous-réseau D, et de là vers la passerelle sur la dorsale. L'opération des NLE au niveau des appareils que la NPDU traverse (énumérés dans l'ordre) se présente comme suit:

a) la NLE de l'appareil E/S passe à sa DLE locale sa propre adresse DL16Address comme étant l'adresse source et l'adresse DL16Address de la passerelle comme étant l'adresse de destination finale. Si le ContractTable spécifie que le ContractID a besoin d'être inclus dans la NPDU, l'en-tête activé par contrat est utilisé; autrement, l'en-tête de base est utilisé si la compression utilisée par le transport le permet (voir 10.5.2.1). Si la taille de la NPDU est plus grande que la taille maxDSDU, la NPDU est fragmentée. La NPDU complète (ou l'ensemble de fragments des NPDU) est alors passée sous forme de(s) DSDU à la DLE associée;

b) la DLE achemine la DSDU vers le routeur dorsal. S'il y a eu fragmentation en a), l'ensemble de fragments est réassemblé en la NPDU au niveau du routeur dorsal. La NLE au routeur dorsal reçoit la NPDU et détermine que la NPDU n'est pas destinée au routeur dorsal, car l'adresse de destination finale dans la primitive DD-Data.indication est l'adresse DL16Address de la passerelle. Le routeur dorsal convertit cette adresse DL16Address en adresse IPv6Address de la passerelle désirée, utilise sa table de routage pour déterminer l'adresse de prochain saut pour atteindre la passerelle et crée un en-tête complet (format montré dans le Tableau 216);

c) la NPDU reconstituée avec l'en-tête de réseau étendu est présentée à l'interface de la dorsale. L'interface de la dorsale achemine la NPDU vers sa destination finale. Dans cet exemple, le prochain saut est la destination finale (la passerelle);

d) la NPDU arrive à la NLE de la passerelle sur la dorsale. La NLE à la passerelle détermine que l'adresse de destination finale est égale à l'adresse de la passerelle elle-même et passe la NSDU à sa TLE.

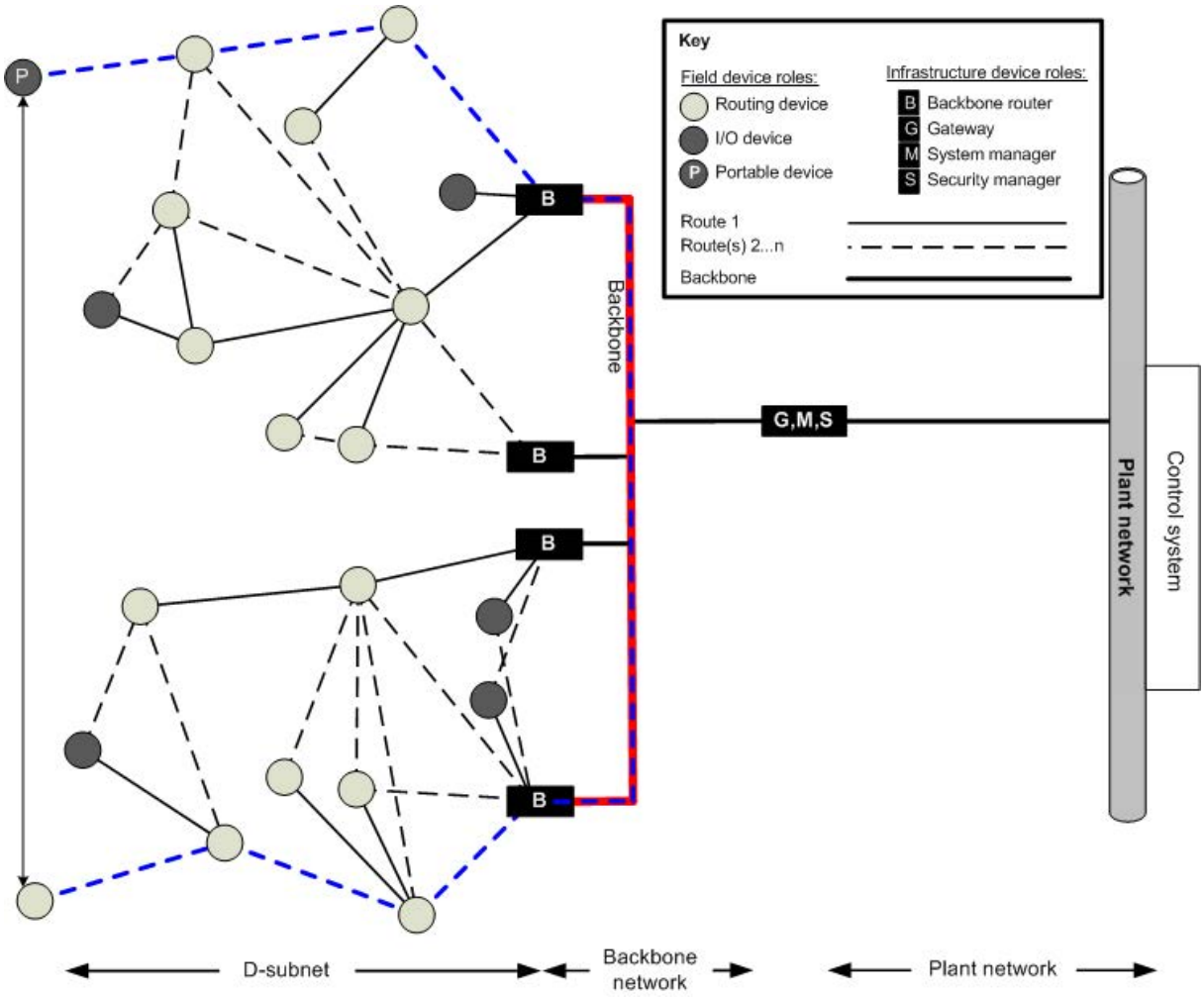


Anglais	Français
I/O device	Appareil E/S
Application	Application
WISN transport	Transport WISN
WISN network	Réseau WISN
WISN data link	Liaison de données WISN
WISN PHY	Physique de WISN
Routing device(s)	Appareil(s) de routage
Backbone router	Routeur dorsal
Backbone network interface	Interface de réseau dorsal
WISN gateway	Passerelle WISN
WISN app	App WISN
Translator	Convertisseur
Control app	App de commande
Plant network transport	Transport de réseau d'installation
Plant network network	Réseau de réseau d'installation
Plant network data link	Liaison de données de réseau d'installation
Plant network PHY	PHY de réseau d'installation
D-subnet	Sous-réseau D
Backbone	Dorsale
Wireless industrial sensor network (WISN)	Réseau de capteurs industriels sans fil (WISN)
Plant network	Réseau d'installation

Figure 101 – Diagramme de suites de protocoles pour acheminer une APDU à partir d'un appareil de terrain vers une passerelle en passant par un routeur dorsal

10.2.7.3 Routage allant d'un appareil de terrain vers un autre appareil de terrain sur un sous-réseau D différent

La Figure 102 montre le routage allant d'un appareil E/S sur un sous-réseau D vers un autre appareil E/S sur un sous-réseau D différent.



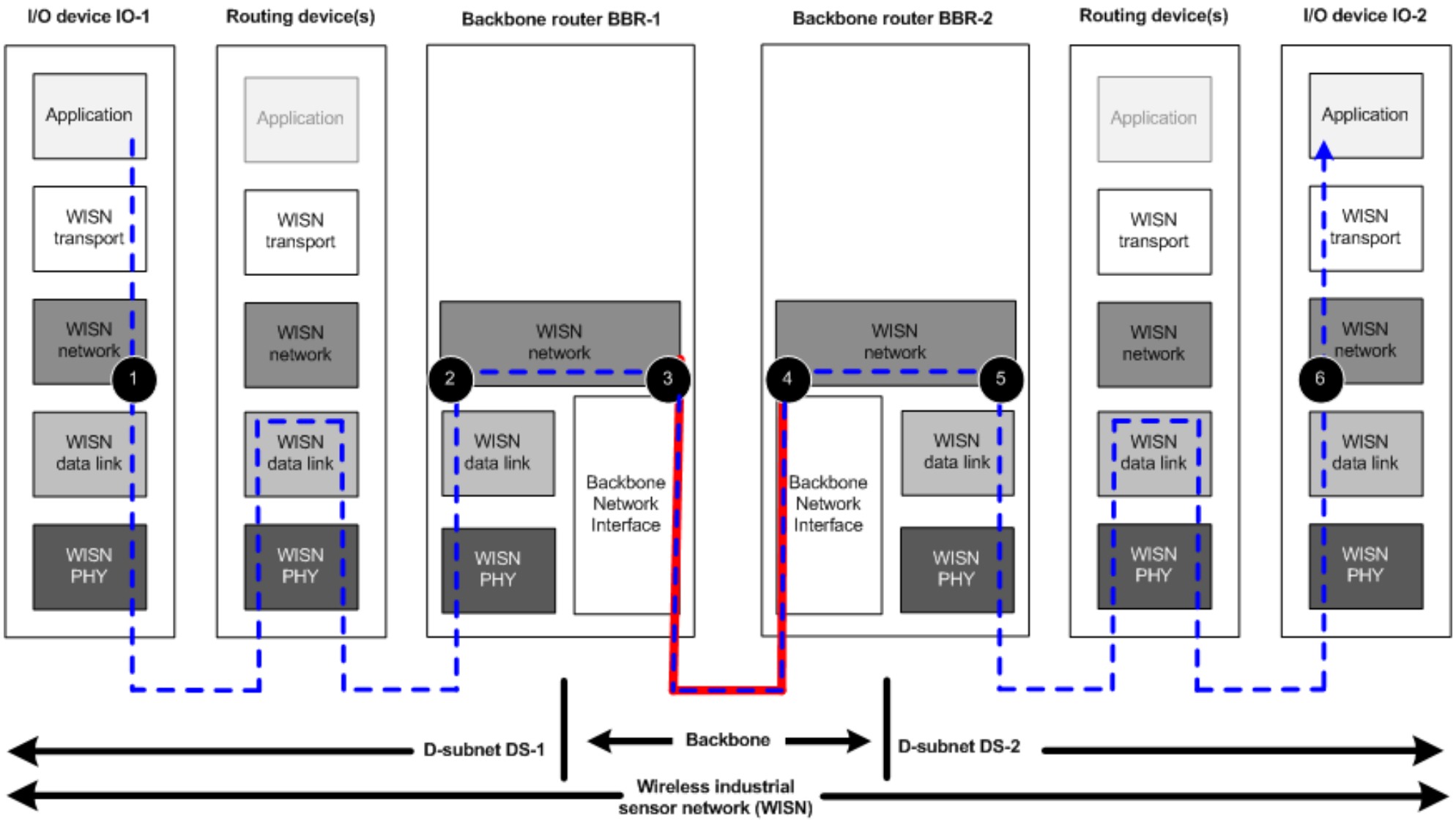
Légende

Anglais	Français
Key	Légende
Field device roles	Rôles d'appareil de terrain
Routing device	Appareil de routage
I/O device	Appareil E/S
Portable device	Appareil portatif
Infrastructure device roles	Rôles d'appareil d'infrastructure
B Backbone router	B Routeur dorsal
G Gateway	G Passerelle
M System manager	M Gestionnaire de système
S Security manager	S Gestionnaire de sécurité
Route 1	Chemin 1
Route(s) 2…n	Chemin(s) 2…n
Backbone	Dorsale
Control system	Système de commande

Anglais	Français
Plant network	Réseau d'installation
D-subnet	Sous-réseau D
Backbone network	Réseau dorsal
Plant network	Réseau d'installation

Figure 102 – Routage allant d'un appareil de terrain sur un sous-réseau D vers un autre appareil de terrain sur un sous-réseau D différent

La Figure 103 montre le flux d'une NPDU entre deux appareils de terrain sur des sous-réseaux D différents (voir 10.2.7.3). Il est supposé que la NPDU n'a besoin d'aucune fragmentation.



Légende

Anglais	Français
I/O device IO-1	Appareil E/S ES-1

Anglais	Français
Application	Application
WISN transport	Transport WISN
WISN network	Réseau WISN
WISN data link	Liaison de données WISN
WISN PHY	PHY de WISN
Routing device(s)	Appareil(s) de routage
Backbone router BBR-1	Routeur dorsal BBR-1
Backbone network interface	Interface de réseau dorsal
Backbone router BBR-2	Routeur dorsal BBR-2
I/O device IO-2	Appareil E/S ES-2
D-subnet DS-1	Sous-réseau D DS-1
Backbone	Dorsale
D-subnet DS-2	Sous-réseau D DS-2
Wireless industrial sensor network (WISN)	Réseau de capteurs industriels sans fil (WISN)

Figure 103 – Diagramme de suites de protocoles pour le routage allant d'un appareil E/S sur un sous-réseau D vers un autre appareil E/S sur un sous-réseau D différent

La NPDU est acheminée sur la dorsale à partir d'un sous-réseau D vers l'autre. Les opérations accomplies par les NLE des appareils à mesure que la NPDU se déplace de l'appareil E/S d'origine (E/S-1) situé dans le sous-réseau D DS-1 vers l'appareil E/S de destination (E/S-2) situé dans le sous-réseau D DS-2 sont comme suit:

a) la NLE au niveau de l'E/S-1 crée la NPDU en utilisant l'en-tête de réseau activé par contrat. La NLE passe la NPDU à la DLE associée comme une DSDU, accompagnée de l'adresse DL16Address de l'E/S-1 au sein de DS-1 comme étant l'adresse de source, et l'adresse DL16Address de l'E/S-2 au sein de DS-1 comme étant l'adresse de destination finale, par l'intermédiaire de la primitive hypothétique DD-DATA.request. Le ContractID est placé dans le champ FlowLabel de l'en-tête activé par contrat;

b) la(les) DPDU qui en résulte(nt) est(sont) acheminée(s) sur DS-1 et arrive(nt) à la DLE de BBR-1, à savoir, le routeur dorsal dans DS-1. Les charges utiles de DPDU sont utilisées pour régénérer la NPDU, qui est vérifiée pour voir si elle est destinée au BBR-1 lui-même. Comme elles ne sont pas destinées au BBR-1, les adresses DL16Addresses de l'E/S-1 et de l'E/S-2 dans la primitive hypothétique DD-DATA.indication sont converties en leurs adresses IPv6Addresses;

c) l'en-tête étendu (au format défini dans le Tableau 216) est créé et présenté à l'interface de la dorsale. Le prochain saut pour cette NPDU sur la dorsale est déterminé en recherchant l'adresse IPv6Address de la destination finale dans la table RT. La RT retourne l'adresse IPv6Address du BBR-2 (le routeur dorsal de DS-2) comme étant le prochain saut, car BBR-2 est le routeur dorsal desservant l'E/S -2. Le ContractID est placé dans le champ FlowLabel de l'en-tête étendu. La priorité de la PDU est placée dans le champ TrafficClass;

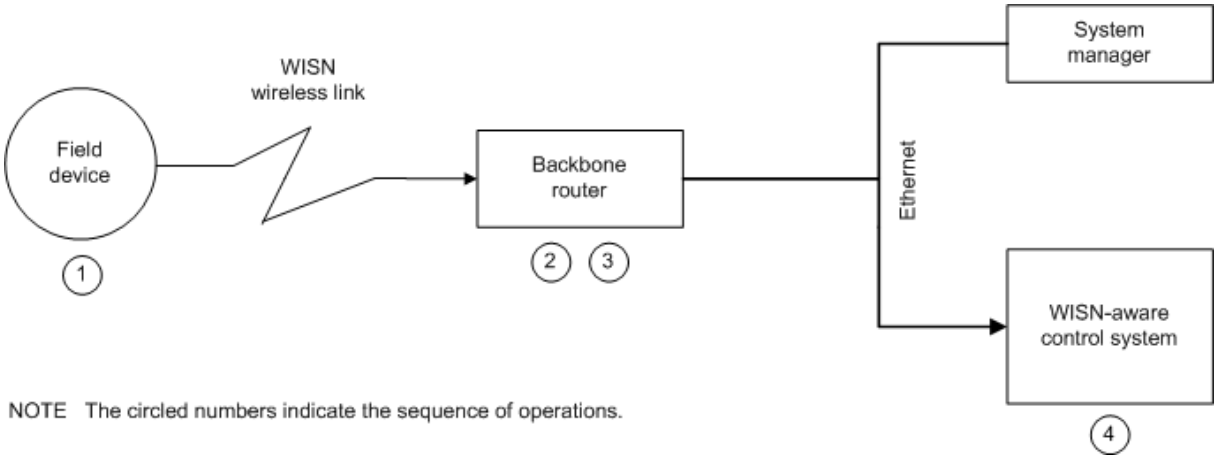
d) la NLE au niveau du BBR-2 reçoit la NPDU sur la dorsale. La NPDU indique que la destination finale est l'adresse IPv6Address de l'E/S-2. Cette NPDU est alors préparée pour le routage sur DS-2 pour atteindre l'E/S-2;

e) la NLE au niveau du BBR-2 crée un en-tête de base de NPDU. Dans la primitive hypothétique consécutive DD-DATA.request, l'adresse DL16Address de l'E/S -1 dans DS-2 est l'adresse d'origine et l'adresse DL16Address de l'E/S-2 dans DS-2 est la destination finale. Le ContractID, extrait du champ FlowLabel de l'en-tête étendu, et la priorité, extraite de TrafficClass, sont également passés à la DL pour permettre la sélection des ressources appropriées de routage (GraphID, etc.);

f) la NPDU arrive à la NLE de l'E/S-2. Comme la destination finale indiquée dans la primitive hypothétique DD-DATA.indication est l'adresse DL16Address de l'E/S -2 dans DS-2, la NLE convertit les adresses en une adresse IPv6Address et passe la NSDU à la TL de l'E/S-2.

10.2.7.4 Exemple de routage sur un réseau dorsal Ethernet

La Figure 104 est un exemple de mise en œuvre du diagramme de suites de protocoles montrée à la Figure 103. Dans ce réseau, un appareil de terrain communique avec un système de commande qui est compatible avec le protocole natif de la présente norme; le réseau dorsal dans cet exemple est un réseau Ethernet transportant le trafic IPv6.



Légende

Anglais	Français
Field device	Appareil de terrain
WISN wireless link	Liaison sans fil WISN
Backbone router	Routeur dorsal
Ethernet	Ethernet
System manager	Gestionnaire de système
WISN-aware control system	Système de commande compatible WISN
NOTE The circled numbers indicate the sequence of operations	NOTE Les nombres entourés indiquent la séquence des opérations

Figure 104 – Exemple de routage sur un réseau dorsal Ethernet

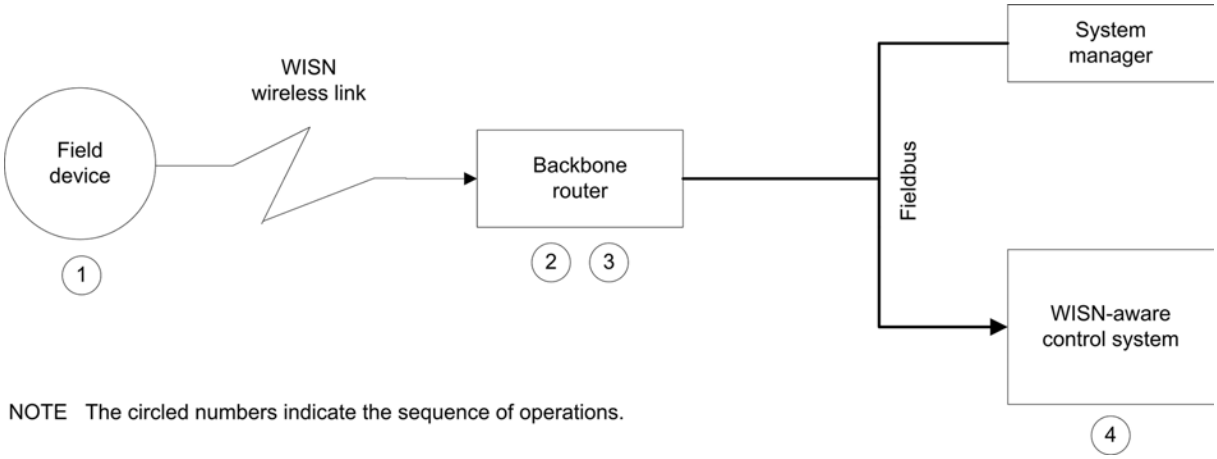
Les cercles numérotés à la Figure 104 indiquent les étapes du processus de routage et l'endroit où elles se produisent:

1) la NLE au niveau de l'appareil de terrain crée une NPDU et la remet à la DLE associée pour émission vers le prochain saut de NL (routeur dorsal). L'adresse de destination finale de la DPDU est l'adresse DL16Address pour le système de commande compatible avec le protocole natif. La maille de DL livre la NPDU à la NLE du routeur dorsal;
2) le routeur dorsal reçoit la NPDU, convertit les adresses DL16Addresses en adresses IPv6Addresses correspondantes et étend la NPDU en une NPDU IPv6 complète;
3) le routeur dorsal envoie la NPDU IPv6 sur l'interface d'Ethernet;
4) la NLE au niveau du système de commande reçoit la NPDU IPv6 issue de son interface d'Ethernet, accomplit le traitement de la NL et passe la NSDU à la TLE.

10.2.7.5 Exemple de routage sur un réseau dorsal

La Figure 105 est une variante de la Figure 104 qui substitue un bus de terrain générique au réseau dorsal Ethernet. Dans cette variante, la NPDU IPv6 est encapsulée à des fins de transport sur le réseau de bus de terrain via une ou plusieurs PDU de bus de terrain.

NOTE Il existe d'autres variantes de ce scénario de réseau dorsal de bus de terrain.



NOTE The circled numbers indicate the sequence of operations.

Légende

Anglais	Français
Field device	Appareil de terrain
WISN wireless link	Liaison sans fil WISN
Backbone router	Routeur dorsal
Fieldbus	Bus de terrain
System manager	Gestionnaire de système
WISN-aware control system	Système de commande compatible WISN
NOTE The circled numbers indicate the sequence of operations	NOTE Les nombres entourés indiquent la séquence des opérations

Figure 105 – Exemple de routage sur un réseau dorsal de bus de terrain

Les cercles numérotés à la Figure 105 indiquent les étapes du processus de routage et l'endroit où elles se produisent:

1) la NLE au niveau de l'appareil de terrain crée une NPDU et la remet à la DLE associée pour émission vers le prochain saut de NL (routeur dorsal). L'adresse de destination finale de la DPDU est l'adresse DL16Address pour le système de commande compatible avec le protocole natif. La maille de DL livre la NPDU à la NLE du routeur dorsal;
2) le routeur dorsal reçoit la NPDU, convertit les adresses DL16Addresses en adresses IPv6Addresses correspondantes et étend la NPDU en une NPDU IPv6 complète;
3) le routeur dorsal encapsule la NPDU entière dans une ou plusieurs PDU de bus de terrain adressées au système de commande;
4) le système de commande (passerelle) reçoit la (ou les) PDU de bus de terrain, extrait la NPDU, accomplit le traitement de la NL et livre la NSDU à la TLE associée.

10.3 Services de données de NLE

10.3.1 Généralités

La TLE utilise l'interface NDSAP de la NLE pour envoyer et recevoir des données. Cette interface est interne à un appareil conforme et est donc hypothétique et ne peut pas être soumise à essai. Les NSAP internes de l'appareil sont décrits à la Figure 16.

Toutes les interfaces entre la NLE et ses NME ou la TLE adjacente et la DLE sont des interfaces internes au sein de l'appareil, et sont donc inobservables. Par conséquent, elles ne sont pas soumises à une normalisation. Ainsi, toute cette description est hypothétique.

10.3.2 N-Data.request

10.3.2.1 Généralités

La primitive N-DATA.request est utilisée par une TLE pour demander à la NLE d'émettre une TSDU.

10.3.2.2 Sémantique

La sémantique de la primitive N-DATA.request se présente comme suit:

```
N-DATA.request   (
                 DestAddress,
                 ContractID,
                 Priority,
                 DE,
                 NSDU,
                 NSDUSize,
                 NSDUHandle,
                 ECN
                 )
```

Le Tableau 203 spécifie les éléments pour la primitive N-DATA.request.

Tableau 203 – Eléments pour la primitive N-Data.request

Nom du type de données normalisé: N-Data.request		
Nom de l'élément	**Identificateur de l'élément**	**Type scalaire de l'élément**
DestAddress (l'adresse IPv6Address de la NLE de destination pour la NSDU)	1	Type: Adresse de l'appareil
ContractID (l'ID de contrat associé aux ressources devant être utilisées pour émettre cette NPDU; cet ID est passé directement à la DLE)	2	Type: Unsigned16 Valeurs nommées: 0 indique l'absence de contrat
Priority (priorité du message au sein du contrat) [a]	3	Type: Unsigned2
DE (indique si la PDU est éligible au rejet)	4	Type: Unsigned1
NSDU (le jeu d'octets formant la NSDU devant être émise par la NL, y compris les en-têtes de transport)	5	Type: OctetString
NSDUSize (le nombre d'octets dans la NSDU devant être émise)	6	Type: Unsigned16
NDSUHandle (le descripteur associé à la NSDU devant être émise)	7	Type: Unsigned16
ECN (notification d'encombrement explicite)	8	Type: Unsigned2
[a] La NLE doit combiner cette priorité avec la priorité de contrat à 2 bits pour coder la priorité D sous la forme d'un champ de 4 bits avant de la passer à la DLE.		

10.3.2.3 Utilisation appropriée

La TLE invoque N-DATA.request pour demander que la NLE émette une NSDU.

10.3.2.4 Effet à la réception

A la réception de N-DATA.request, la NLE construit les en-têtes de NL de NPDU, d'abord en élidant le premier octet du LoWPAN_NHC de la NSDU si l'en-tête de base est utilisé, puis en fragmentant la NPDU (s'il y a lieu), et en l'acheminant vers la DLE pour émission sur le sous-réseau D local. Si ContractID est égal à zéro, la NLE traite la NSDU comme étant une PDU relative au rattachement avec une adresse EUI64Address de destination associée, pour

laquelle l'adresse IPv6Address de destination exigée est dérivée de l'adresse locale à une liaison passée sous forme du paramètre DestAddress.

10.3.3 N-Data.confirm

10.3.3.1 Généralités

La primitive N-DATA.confirm est utilisée pour rapporter le résultat d'une primitive N-DATA.request.

10.3.3.2 Sémantique

La sémantique de la primitive N-DATA.confirm se présente comme suit:

```
N-Data.confirm    (
                  NSDUHandle,
                  status
                  )
```

Le Tableau 204 spécifie les éléments pour la primitive N-DATA.confirm.

Tableau 204 – Eléments pour la primitive N-Data.confirm

Nom de l'élément	Identificateur de l'élément	Type scalaire de l'élément
NSDUHandle (le descripteur de la NSDU dont le statut est rapporté)	1	Type: Unsigned16
Status (le résultat de la primitive N-DATA.request qui a acheminé la NSDU)	2	Type: Unsigned Valeurs nommées: 0: réussite

10.3.3.3 Moment de sa génération

La NLE génère une primitive N-DATA.confirm comme réponse différée à une primitive N-DATA.request. La primitive N-DATA.confirm retourne un statut vers la TLE qui indique soit SUCCESS (réussite), soit FAILURE (échec).

10.3.3.4 Utilisation appropriée

La primitive N-DATA.confirm notifie à la TLE le résultat de sa demande d'émission d'une NSDU.

10.3.4 N-Data.indication

10.3.4.1 Généralités

La primitive N-DATA.indication est utilisée par une NLE pour livrer à une TLE associée une TSDU reçue.

10.3.4.2 Sémantique

La sémantique de la primitive N-DATA.indication se présente comme suit:

```
N-Data.indication (
                  SrcAddress,
                  DestAddress,
                  NSDU,
                  NSDUSize,
                  ECN,
                  Priority
                  )
```

Le Tableau 205 spécifie les éléments pour la primitive N-DATA.indication.

Tableau 205 – Eléments pour la primitive N-Data.indication

Nom de l'élément	Identificateur de l'élément	Type scalaire de l'élément
SrcAddress (l'adresse IPv6Address de la source de la NSDU)	1	Type: Adresse de l'appareil
DestAddress (l'adresse IPv6Address de la destination de la NSDU, par exemple la propre adresse IPv6Address de l'appareil)	2	Type: Adresse de l'appareil
NSDU (la NSDU reçue, y compris les en-têtes de TL associés)	3	Type: Séquence d'octets
NSDUSize (le nombre d'octets dans la NSDU reçue)	4	Type: Unsigned16
ECN (informations relatives à la notification d'encombrement explicite issues de la NSDU reçue)	5	Type: BitArray4
Priority (priorité de NPDU de 4 bits telle que reçue)	6	Type: Unsigned4

10.3.4.3 Utilisation appropriée

La NLE invoque la primitive N-DATA.indication pour notifier à la TLE associée une NSDU reçue et les informations associées relatives à l'acheminement. Si la NPDU reçue contient un en-tête de base, alors, avant de passer la NSDU à la TLE, la NLE restaure (ajoute en préfixe) le premier octet de la NSDU comme LoWPAN_NHC (= 0x 1111 0111), qui avait été élidé lorsque l'en-tête de base avait été construit. Si l'adresse D source reçue en provenance de la DLE sous-jacente est une adresse EUI64Address, une adresse IPv6Address locale à une liaison est construite à partir de l'adresse EUI64Address et elle est passée comme paramètre SrcAddress de la primitive N-DATA.indication. Le paramètre DestAddress de la primitive N-DATA.indication est l'adresse IPv6Address locale à une liaison de l'appareil lorsqu'elle est utilisée pour les APDU relatives au rattachement, ou l'adresse IPv6Address globalement assignée pour le fonctionnement ultérieur au rattachement.

10.3.4.4 Effet à la réception

A la réception d'une primitive N-DATA.indication, la TLE peut traiter la NSDU rapportée.

10.4 Objet de gestion de NL

10.4.1 Base d'informations de gestion de NL

Le Tableau 206 spécifie les attributs de l'objet de gestion de NL (NLMO).

Tableau 206 – Attributs de NLMO *(1 de 4)*

Nom du type d'objet normalisé: NL management object (NLMO, objet de gestion de NL)				
Identificateur du type d'objet normalisé: 123				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Type d'attribut**	**Description du comportement de l'attribut**
Backbone_Capable	1	Indicateur booléen indiquant si l'appareil est à capacité de dorsale	Type: Boolean1 Classification: Static Accessibilité: Lecture seule	Valeur fixe basée sur les capacités de l'appareil et sur les détails de mise en œuvre. La capacité de dorsale peut être omise par un gestionnaire de système
DL_Capable	2	Indicateur booléen indiquant si l'appareil est capable de communiquer sur le support de Type A sans fil de la présente norme	Type: Boolean1 Classification: Static Accessibilité: Lecture seule	Valeur fixe basée sur les capacités de l'appareil et sur les détails de mise en œuvre. La capacité d'interface de DL peut être omise par un gestionnaire de système
DL16Address	3	L'adresse DL16Address de l'appareil	Type: DL16Address Classification: Static Accessibilité: Lecture/écriture Plage valide: 1.. 2^{15} -1	Une valeur fixe telle qu'assignée par le gestionnaire de système au moment du rattachement.[a] Cet attribut est un doublon des attributs correspondants dans le DMO et le DLMO
Long_Address	4	L'adresse IPv6Address de l'appareil	Type: IPv6Address Classification: Static Accessibilité: Lecture/écriture Plage valide: toutes les valeurs avec réglage sur bit de poids fort	Une valeur fixe telle qu'assignée par le gestionnaire de système au moment du rattachement.[a] Cet attribut est un doublon des attributs correspondants dans le DMO et le DLMO
Route_Table	5	La table de routage qui inclut les informations pour acheminer une NPDU	Type: Array of NLRouteTbl structures (voir NLRouteTbl) Classification: Static Accessibilité: Lecture/écriture Plage valide: Voir Tableau 208	La table de routage est constituée d'une adresse de destination, d'un prochain saut de réseau pour la destination en question et du nombre de sauts de réseau nécessaires. La structure de la table de routage et ses alertes et méthodes correspondantes sont débattues en 10.2.6.2. La taille de la table de routage présente dans des appareils qui fonctionnent seulement au sein du sous-réseau D et ne sont pas à capacité de dorsale est présentée dans le Tableau B.18
Enable_Default_Route	6	Une valeur booléenne mise pour indiquer si un routage par défaut est activé	Type: Boolean1 Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: Désactivé	Permet un chemin par défaut

Tableau 206 *(2 de 4)*

Nom du type d'objet normalisé: NL management object (NLMO, objet de gestion de NL)				
Identificateur du type d'objet normalisé: 123				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Type d'attribut**	**Description du comportement de l'attribut**
Default_Route_Entry	7	Adresse de destination associée au chemin par défaut	Type: IPv6Address Classification: Static Accessibilité: Lecture/écriture	L'attribut peut être utilisé pour rechercher le chemin par défaut dans la table de routage. Des paquets qui incluent une adresse de destination sans l'entrée correspondante dans la table de routage doivent être transmis en utilisant le chemin par défaut. Les appareils de terrain peuvent utiliser le chemin par défaut pour envoyer des paquets vers des appareils fonctionnant sur la dorsale qui n'ont pas d'adresse assignée de 16 bits mais seulement des adresses IPv6Addresses
Contract_Table	8	Inclut des informations pour construire l'en-tête de réseau pour un flux de PDU particulier	Type: Array of NLContractTbl structures (voir NLContractTbl) Classification: Static Accessibilité: Lecture/écriture Plage valide: Voir Tableau 207	La table de contrat consiste en l'adresse de destination, priorité pour un flux particulier. La table inclut également des informations indiquant si, oui ou non, un ContractID a besoin d'être transporté dans la NPDU. La portée d'un ID de contrat est locale à un appareil, mais la combinaison d'une adresse de source (128 bits) et du ContractID est unique au niveau global. La structure de la table de contrat et ses alertes et méthodes correspondantes sont débattues ci-dessous
Address_Translation_Table	9	Inclut l'adresse IPv6Address et le pseudonyme DL16Address pour les adresses IPv6Addresses	Type: Array of NLATTbl structures (voir NLATTbl) Classification: Static Accessibilité: Lecture/écriture Plage valide: Voir Tableau 209	La table de conversion d'adresse dans un appareil est indexée par l'adresse IPv6Address et stocke le pseudonyme DL16Address correspondant dans le sous-réseau D auquel l'appareil appartient

Tableau 206 *(3 de 4)*

Nom du type d'objet normalisé: NL management object (NLMO, objet de gestion de NL)				
Identificateur du type d'objet normalisé: 123				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Type d'attribut**	**Description du comportement de l'attribut**
Max_NSDU_size	10	Taille maximale de l'unité de données de service prise en charge par la NL de l'appareil	Type: Unsigned16 Classification: Constant	Valeur fixe basée sur les capacités mémoire de l'appareil et sur les détails de mise en œuvre. La valeur 1 280 est issue de l'IETF RFC 4944. Voir 6.2.8 concernant la façon dont cette valeur peut nécessiter une fragmentation au niveau de l'AL. Les NSDU qui dépassent la valeur maximale peuvent être rejetées par l'appareil
			Accessibilité: Lecture seule	
			Valeur par défaut: 70	
			Plage valide: 70..1 280	
Frag_Reassembly_Timeout	11	Quantité de temps (en secondes) dont a besoin un tampon de réassemblage pour être maintenu ouvert afin que les fragments y pénètrent avant de rejeter la NPDU	Type: Unsigned16 Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: 60 Plage valide: 1..600	La valeur par défaut est 60 s, mais le gestionnaire de système peut changer cette valeur
Frag_Datagram_Tag	12	Numéro d'étiquette actuel pour la fragmentation au niveau de l'appareil	Type: Unsigned16 Classification: Dynamic Accessibilité: Lecture seule Valeur par défaut: Aléatoire uniforme	Un nouveau numéro d'étiquette est utilisé pour chaque NPDU qui a besoin d'être fragmentée. Le numéro d'étiquette est incrémenté de 1 pour chaque NPDU devant être fragmentée. La valeur revient à zéro après 65 535
NLRouteTblMeta	13	Métadonnées pour l'attribut Route Table (attribut 5)	Type: Metadata_attribute Classification: Static Accessibilité: Lecture seule	Métadonnées contenant un compte du nombre d'entrées dans la table et la capacité (le nombre total de rangées autorisées) pour cette table. La taille de la table de routage présente dans des appareils qui fonctionnent seulement au sein du sous-réseau D et ne sont pas à capacité de dorsale est présentée dans le Tableau B.18

Tableau 206 *(4 de 4)*

Nom du type d'objet normalisé: NL management object (NLMO, objet de gestion de NL)				
Identificateur du type d'objet normalisé: 123				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Type d'attribut**	**Description du comportement de l'attribut**
NLContractTblMeta	14	Métadonnées pour l'attribut Contract Table (attribut 8)	Type: Metadata_attribute Classification: Static Accessibilité: Lecture seule	Métadonnées contenant un compte du nombre d'entrées dans la table et la capacité (le nombre total de rangées autorisées) pour cette table.
NLATTblMeta	15	Métadonnées pour l'attribut Address Translation Table (attribut 9)	Type: Metadata_attribute Classification: Static Accessibilité: Lecture seule	Métadonnées contenant un compte du nombre d'entrées dans la table et la capacité (le nombre total de rangées autorisées) pour cette table.
DroppedNPDUAlertDescriptor	16	Décrit comment une alerte NPDU abandonnée est rapportée sur le réseau	Type: Alert report descriptor Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: [true, 7]	-

[a] L'attribut doit rester inchangé au cours du fonctionnement normal; seul un gestionnaire de système peut le modifier. L'assignation d'une nouvelle adresse DL16Address doit déclencher un Join_Command = 3, restartAsProvisioned, à la suite de quoi l'appareil doit effectuer un nouveau rattachement au réseau sans fil.

10.4.2 Bases d'informations de gestion structurées

Le NLMO définit trois bases d'informations de gestion structurées (SMIB) sous forme de tables. Ce sont la table de contrat, la table de routage (RT) et la table de conversion d'adresse (ATT).

Le Tableau 207 spécifie les éléments pour la table de contrat. Les appareils qui ne sont pas à capacité de dorsale peuvent élider le champ Source_Address.

Tableau 207 – Structure de la table de contrat

Nom du type de données normalisé: NLContractTbl		
Code du type de données normalisé: 441		
Nom de l'élément	**Identificateur de l'élément**	**Type scalaire de l'élément**
Contract_ID*	1	Type: Unsigned16 Classification: Static Accessibilité: Lecture/écriture
Source_Address*	2	Type: IPv6Address Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: 0
Destination_Address	3	Type: IPv6Address Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: 0
Contract_Priority	4	Type: Unsigned2 Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: 00
Include_Contract_Flag	5	Type: Boolean1 Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: false
NOTE * indique un champ d'indice		

Le Tableau 208 spécifie les éléments pour la table de routage.

Tableau 208 – Eléments pour la table de routage

Nom du type de données normalisé: NLRouteTbl		
Code du type de données normalisé: 442		
Nom de l'élément	**Identificateur de l'élément**	**Type scalaire de l'élément**
Destination_Address*	1	Type: IPv6Address Classification: Static Accessibilité: Lecture/écriture
Next_Hop	2	Type: IPv6Address Classification: Static Accessibilité: Lecture/écriture
NWK_HopLimit	3	Type: Unsigned8 Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: 64
Outgoing_Interface	4	Type: Unsigned1 Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: 0 Valeurs nommées: 0: DL; 1: Dorsale
NOTE * indique un champ d'indice		

Le Tableau 209 spécifie les éléments pour la table de conversion d'adresse.

Tableau 209 – Structure de la table de conversion d'adresse

Nom du type de données normalisé: NLATTbl		
Code du type de données normalisé: 443		
Nom de l'élément	**Identificateur de l'élément**	**Type scalaire de l'élément**
Long_Address*	1	Type: IPv6Address Classification: Static Accessibilité: Lecture/écriture
Short_Address	2	Type: DL16Address Classification: Static Accessibilité: Lecture/écriture
NOTE * indique un champ d'indice		

10.4.3 Méthodes de l'objet de gestion de NL

Les méthodes normalisées telles que la lecture (read) et l'écriture (write) peuvent être utilisées pour les MIB scalaires ou structurées dans leur intégralité. Ces méthodes sont utilisées pour manipuler des tables. Elles permettent l'accès à une rangée particulière d'une MIB structurée en fonction d'un champ indice unique. Tous les appareils doivent être capables de manipuler immédiatement les attributs de NLMO. Les appareils prennent en charge la manipulation différée de ces attributs en utilisant des méthodes de basculement, mais ne sont pas tenus de le faire.

Il est supposé que les tables ont un champ indice unique, qui peut être soit un seul élément, soit la concaténation de plusieurs éléments. Le champ indice est supposé être le(s) (quelque(s)) premier(s) élément(s) de la table ou la concaténation des quelques premiers éléments de la table. Par exemple, le champ indice de la table de contrat est la concaténation du Contract_ID et Source_Address.

Ces méthodes ne spécifient pas l'interface entre la NME et le NLMO, car cette interface est interne à un appareil particulier. L'entité de gestion peut garder des copies locales des MIB.

Le Tableau 210 décrit les méthodes pour la manipulation des MIB structurées. Ces méthodes sont basées sur les modèles Read_Row, Write_Row, et Delete_Row définis en Annexe J.

Tableau 210 – Méthodes de manipulation des MIB structurées du NLMO

Nom du type d'objet normalisé: NL management object (NLMO, objet de gestion de NL)		
Identificateur du type d'objet normalisé: 123		
Nom de la méthode	**ID de la méthode**	**Description de la méthode**
Set_row_RT	1	Méthode pour établir (soit ajouter, soit modifier) la valeur d'une seule rangée de la table de routage. La méthode utilise le modèle de méthode Write_Row défini à l'Annexe J avec les arguments suivants: Attribute_ID: 5 (table de routage) Index 1: 1 (Destination_address)
Get_row_RT	2	Méthode pour obtenir la valeur d'une seule rangée de la table de routage. La méthode utilise le modèle de méthode Read_Row défini à l'Annexe J avec les arguments suivants: Attribute_ID: 5 (table de routage) Index 1: 1 (Destination_address)
Delete_row_RT	3	Méthode pour supprimer une seule rangée de la table de contrat. La méthode utilise le modèle de méthode Delete_Row défini à l'Annexe J avec les arguments suivants: Attribute_ID: 5 (table de routage) Index 1: 1 (Destination_address)
Set_row_ContractTable	4	Méthode pour établir (soit écrire, soit modifier) la valeur d'une seule rangée de la table de contrat. La méthode utilise le modèle de méthode Write_Row défini à l'Annexe J avec les arguments suivants: Attribute_ID: 8 (table de contrat) Index 1: 1 (ContractID) Index 2: 2 (Source Address)
Get_row_ContractTable	5	Méthode pour obtenir la valeur d'une seule rangée de la table de contrat. La méthode utilise le modèle de méthode Read_Row défini à l'Annexe J avec les arguments suivants: Attribute_ID: 8 (table de contrat) Index 1: 1 (ContractID) Index 2: 2 (Source Address)
Delete_row_ContractTable	6	Méthode pour supprimer la valeur d'une seule rangée de la table de contrat. La méthode utilise le modèle de méthode Delete_Row défini à l'Annexe J avec les arguments suivants: Attribute_ID: 8 (table de contrat) Index 1: 1 (ContractID) Index 2: 2 (Source Address)
Set_row_ATT	7	Méthode pour établir (soit ajouter, soit modifier) la valeur d'une seule rangée de la table de conversion d'adresse. La méthode utilise le modèle de méthode Write_Row défini à l'Annexe J avec les arguments suivants: Attribute_ID: 9 (table de conversion d'adresse) Index 1: 1 (Long Address)
Get_row_ATT	8	Méthode pour obtenir la valeur d'une seule rangée de la table de conversion d'adresse. La méthode utilise le modèle de méthode Read_Row défini à l'Annexe J avec les arguments suivants: Attribute_ID: 9 (table de conversion d'adresse) Index 1: 1 (Long Address)

Nom du type d'objet normalisé: NL management object (NLMO, objet de gestion de NL)		
Identificateur du type d'objet normalisé: 123		
Nom de la méthode	**ID de la méthode**	**Description de la méthode**
Delete_row_ATT	9	Méthode pour supprimer une seule rangée de la table de conversion d'adresse. La méthode utilise le modèle de méthode Delete_Row défini à l'Annexe J avec les arguments suivants: Attribute_ID: 9 (table de conversion d'adresse) Index 1: 1 (Long Address)

Le Tableau 211 décrit l'alerte pour indiquer une PDU abandonnée/erreur de PDU.

Tableau 211 – Alerte pour indiquer une PDU abandonnée/erreur de PDU

Nom du type d'objet normalisé: NL management object (NLMO, objet de gestion de NL)					
Identificateur du type d'objet normalisé: 123					
Description de l'alerte: Alerte pour indiquer une PDU abandonnée/erreur de PDU					
Classe d'alertes (Enumerated: alarme ou événement)	**Catégorie d'alertes (Enumerated: diagnostic d'appareil, diagnostic de comm., sécurité ou processus)**	**Type d'alerte (Enumerated: en fonction de la catégorie d'alertes)**	**Priorité d'alertes (Enumerated: haut, moyen, faible, journal seulement)**	**Type de données de la valeur**	**Description de la valeur incluse avec l'alerte**
0 = Event	1 = Comm. diagnostic	0 = NL_Dropped_PDU	7 =Medium	Type: OctetString	La valeur est une chaîne d'octets constituée d'au moins deux octets. Le premier octet (Unsigned8) spécifie la taille du champ valeur inclus avec l'alerte en octets. Le second octet (Unsigned8) achemine la classe de diagnostic. Valeurs nommées: 0: réservé; 1: destination inaccessible; 2: erreur de fragmentation; 3: temporisation de réassemblage; 4: limite de saut atteinte; 5: erreurs d'en-tête; 6: pas de chemin, prochain saut inaccessible; 7: mémoire insuffisante; 8: taille de NPDU trop large; 9..255: réservés. Les octets restants incluent l'en-tête de NPDU pour la PDU abandonnée

10.5 Formats de NPDU

10.5.1 Généralités

Chaque NPDU doit être constituée de deux composantes de base:

- Un en-tête de réseau comportant probablement l'adressage, la classe de service ((CoS), et les champs de fragmentation.
- Une charge utile de réseau de taille variable contenant les données qui ont besoin d'être émises.

La présente norme doit permettre trois formats différents d'en-tête de NPDU:

– l'en-tête de base;

– l'en-tête activé par contrat; et

– l'en-tête complet.

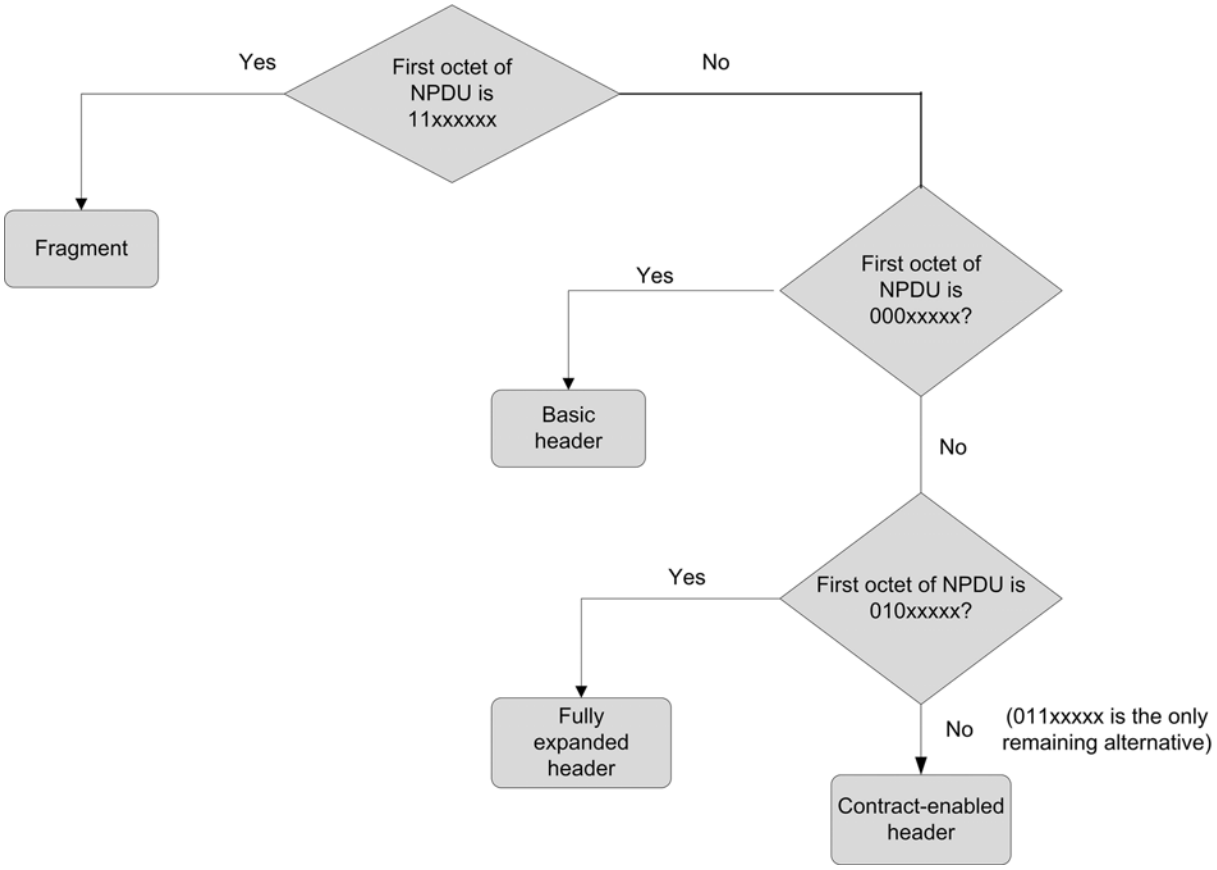
Les appareils conformes à la présente norme doivent utiliser des préfixes d'expédition (les 3 bits de poids faible du premier octet de la NPDU) pour distinguer entre ces formats d'en-tête (voir Figure 106). Le préfixe 000 doit indiquer que le format d'en-tête de NPDU est l'en-tête de base. Le préfixe d'en-tête de base se conforme au NALP dispatch de 6LoWPAN.

Le préfixe 011 doit indiquer l'en-tête activé par contrat. L'en-tête activé par contrat se conforme au LOWPAN_IPHC dispatch de 6LoWPAN.

Le préfixe 010 doit indiquer le format d'en-tête complet. Les appareils à ressources contraintes qui opèrent dans le sous-réseau D sans fil et sont à capacité de dorsale peuvent construire l'en-tête complet, mais ne sont pas tenus de le faire. Si un paquet qui est construit en utilisant l'en-tête complet est reçu par un appareil qui n'est pas à capacité de dorsale, l'appareil peut rejeter la NPDU et envoyer une alerte PDU abandonnée/erreur de PDU avec la valeur 5 indiquant une erreur d'en-tête.

Enfin, un préfixe de 110 ou de 111 doit indiquer que la PDU est un fragment d'une plus grande NPDU qui a besoin d'être réassemblée.

La présente norme utilise principalement des adresses de 16 bits pour la toute première destination (initiateur) et la destination finale dans les DPDU émises sur le réseau sans fil selon l'IEEE 802.15.4. Par conséquent, il convient que l'en-tête de réseau soit au format d'en-tête de base ou au format d'en-tête activé par contrat pour les NPDU émises sur les liaisons sans fil selon l'IEEE 802.15.4. Il convient, bien que cela ne soit pas interdit, de ne pas utiliser le format d'en-tête complet pour l'émission de NPDU sur le moyen de terrain.



Légende

Anglais	Français
Fragment	Fragment
Yes	Oui
First octet of NPDU is 11xxxxxx	Premier octet de la NPDU est 11xxxxxx?
No	Non
First octet of NPDU is 000xxxxx?	Premier octet de la NPDU est 000xxxxx?
Basic header	En-tête de base
First octet of NPDU is 010xxxxx?	Premier octet de la NPDU est 010xxxxx?
Fully expanded header	En-tête étendu complet
(011xxxxx is the only remaining alternative)	(011xxxxx est la seule possibilité restante)
Contract-enabled header	En-tête activé par contrat

Figure 106 – Distinction entre les formats d'en-tête de NPDU

Les profils des bits d'octet dispatch sont montrés dans le Tableau 212.

Tableau 212 – Profils d'en-tête communs

Profil de dispatch	Type d'en-tête
000xxxxx	NALP (NPDU Not A 6LoWPAN) – En-tête de base
010xxxxx	IPv6 (en-tête IPv6 décompressé) – En-tête complet
011xxxxx	LoWPAN_IPHC (en-tête IPv6 compressé) – En-tête activé par contrat
11xxxxxx	LoWPAN fragment

Les en-têtes de NL doivent suivre les formats définis dans le présent document.

10.5.2 Format d'en-tête de base pour NL

10.5.2.1 Usage prévu

La DL de la présente norme utilise une maille de niveau liaison. Dans le cas le plus commun, une PDU traversera un seul sous-réseau D, si bien que l'en-tête de base est optimisé pour réduire au maximum le surdébit de NPDU. Le chemin qui a besoin d'être pris par la PDU est connu de l'appareil d'entrée dans le sous-réseau D; cet appareil d'entrée prend toutes les décisions nécessaires relatives au routage de DL. Le ContractID n'est pas émis dans l'en-tête de réseau de base.

L'en-tête de base pour la NL doit être utilisé seulement si l'en-tête de protocole datagramme d'utilisateur (UDP) est complètement comprimé (à savoir, les numéros de port source et destination sont compressés à quatre (4) bits chacun et la somme de contrôle de l'UDP est élidée). Le NL peut déterminer si, oui ou non, l'en-tête d'UDP est complètement comprimé en regardant l'octet LOWPAN_NHC, qui est toujours le premier octet de la NSDU passée à la NL par la TL. Sachant que l'en-tête de base est utilisé seulement en cas d'en-tête UDP complètement comprimé (à savoir, la valeur fixe et connue de LOWPAN_NHC de l'UDP), l'octet LOWPAN_NHC de l'UDP doit être élidé par la NL d'origine et restauré par la NL de destination.

10.5.2.2 Format

Le Tableau 213 décrit le format d'en-tête de réseau de base.

Tableau 213 – Format d'en-tête de NL de base

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	Dispatch							
(variable)	Charge utile de réseau							

L'en-tête de NL de base doit être constitué d'un seul champ Dispatch comme suit:

- Dispatch: le champ dispatch indique le format d'en-tête de NL. Pour l'en-tête de base, le champ dispatch doit avoir la valeur de 0x 0000 0001.

10.5.2.3 Relation à 6LoWPAN

Le format 6LoWPAN permet à tous les champs de l'en-tête IPv6 complet d'être élidés. Les octets dispatch et de codage pour ce faire sont:0x0111 1110 0111 0111, ce qui indique (lorsqu'analysés comme étant 011.11.1.10.0.1.11.0.1.11):

- Dispatch = 011;
- TF = 11 (classe de trafic et étiquette de flux toutes deux élidées);
- NH = 1 (prochain champ d'en-tête élidé);
- HLIM = 10 (HopLimit = 64);
- CID = 0 (absence d'extension d'identificateur de contexte);
- SAC = 1 (compression d'adresse source avec états; la table ATT fournit le contexte);
- SAM = 11 (adresse source complètement élidée);
- M = 0 (pas de multidiffusion);
- DAC = 1 (compression d'adresse de destination avec états; la table ATT fournit le contexte);
- DAM = 11 (adresse de destination complètement élidée).

Le format 6LoWPAN permet également à l'en-tête d'UDP d'être comprimé, si bien que les numéros de port source et destination sont de quatre (4) bits chacun et la somme de contrôle est élidée. Dans ce cas, le champ LOWPAN_NHC de l'UDP a la valeur 0x 1111 0111, ce qui indique (lorsqu'analysée comme étant 11110.1.11):

– protocole = 11 110 (UDP);
– compression de somme de contrôle = 1 (somme de contrôle élidée);
– compression de port = 11 (les ports source et destination sont compressés en quatre (4) bits chacun et leur préfixe implicite est 0xF0B).

L'en-tête de base est essentiellement une abréviation d'un seul octet pour les trois octets (deux octets d'en-tête IP complètement compressés et un octet de l'en-tête UDP complètement compressé) notés ci-dessus. Comme il s'agit d'une abréviation, elle est totalement compatible avec le format 6LoWPAN. Un appareil recevant une NPDU d'en-tête de base peut étendre l'octet dispatch d'en-tête de base aux trois octets susmentionnés et obtenir une PDU conforme à 6LoWPAN.

10.5.3 Format d'en-tête de réseau activé par contrat

10.5.3.1 Usage prévu

Tout comme l'en-tête de base, l'en-tête de réseau activé par contrat est prévu pour une utilisation au sein de sous-réseaux D. Le tout premier appareil (initiateur) d'une NPDU peut utiliser l'en-tête activé par contrat en lieu et place de l'en-tête de base s'il est souhaitable que les appareils autres que le tout premier appareil (initiateur) aient connaissance du flux de NPDU auquel la NPDU appartient. Par exemple, un routeur intermédiaire sur la dorsale peut avoir besoin de choisir:

- des ressources de dorsale appropriées à la sortie d'un sous-réseau D d'origine; ou
- des ressources de DL appropriées (ID de graphe, priorité, etc.) à l'entrée d'un sous-réseau D de destination.

Le contrat inclut un fanion pour indiquer à la NL d'origine si, oui ou non, l'en-tête de réseau exige l'inclusion du ContractID.

L'en-tête activé par contrat doit également être utilisé si le LOWPAN_NHC de l'UDP n'indique pas la complète compression de l'en-tête UDP. Les messages du processus de rattachement entre le nouvel appareil et le routeur d'annonce s'inscriront toujours dans cette catégorie, car ils n'élident pas la somme de contrôle UDP.

10.5.3.2 Format

Le Tableau 214 décrit le format d'en-tête activé par contrat.

Tableau 214 – Format d'en-tête de NL activé par contrat

<table>
<tr><th rowspan="2">Nombre d'octets</th><th colspan="8">Bits</th></tr>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td rowspan="2">2</td><td colspan="3">Dispatch LOWPAN_IPHC</td><td colspan="5">Codage de LOWPAN_IPHC (bits 8..12)</td></tr>
<tr><td colspan="8">Codage de LOWPAN_IPHC (bits 0..7)</td></tr>
<tr><td rowspan="3">0 ou 3</td><td colspan="4">Reserved</td><td colspan="4">FlowLabel (bits16..19)</td></tr>
<tr><td colspan="8">Flow Label (bits 8..15)</td></tr>
<tr><td colspan="8">Flow Label (bits 0..7)</td></tr>
<tr><td>0..1</td><td colspan="8">HopLimit</td></tr>
<tr><td>(variable)</td><td colspan="8">Charge utile de réseau</td></tr>
</table>

Les champs comprennent:

- Dispatch LOWPAN_IPHC: ce champ doit indiquer que le format d'en-tête est "contract-enabled" (activé par contrat) et que les bits de codage de compression d'en-tête LOWPAN_IPHC suivent. Le champ dispatch LOWPAN_IPHC doit être 011.
- Codage de LOWPAN_IPHC: ce champ a une longueur de 13 bits; sa valeur doit être codée comme étant 0x0 11HH 0111 0111 où les octets 3..5 sont présents pour transporter l'ID de contrat, ou comme étant 0x1 11HH 0111 0111 où les octets 3..5 sont élidés. Dans un cas comme dans l'autre, HH doit avoir la valeur 00 si HopLimit est transporté porté en ligne, 01 si la limite de saut est 1, 10 si la limite de saut est 64 et 11 si la limite de saut est 255.
- FlowLabel: les 16 bits de poids faible de FlowLabel doivent être mis à ContractID. Les 4 bits de poids fort doivent être tous des zéros. Ce champ doit être présent seulement si les octets 3 à 5 sont présents, comme l'indique le codage de LOWPAN_IPHC.
- HopLimit: ce champ doit indiquer le nombre de sauts de la couche 3 autorisés avant que la NPDU ne soit rejetée. Le champ HopLimit doit être mis à une valeur indiquée par la table de routage de l'appareil (RT; voir 10.2.6.2). La valeur par défaut du champ HopLimit doit être 64. Les appareils qui fonctionnent seulement au sein du sous-réseau D et ne sont pas à capacité de dorsale doivent mettre le champ HopLimit à 1. De la perspective de la NL, tout appareil joignable par l'intermédiaire de la maille de DL est à un seul saut de réseau de distance.

Pour les messages du processus de rattachement, le codage de LOWPAN_IPHC doit avoir la valeur 0x1 1101 0111 0111 pour indiquer que les octets 3 à 5 sont élidés (pas d'ID de contrat) et que la limite de saut est 1 (champ HopLimit élidé).

10.5.3.3 Relation à 6LoWPAN

Le Tableau 215 montre le format de codage de 6LoWPAN_IPHC.

Tableau 215 – Format de codage de 6LoWPAN_IPHC

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	0	1	1	TF		NH	HLIM	
1	CID	SAC	SAM		M	DAC	DAM	
…	Champs non compressés							

Le codage pour l'en-tête de NL activé par contrat est dérivé en utilisant les valeurs suivantes pour les champs de codage de 6LoWPAN_IPHC:

- TF = 01 ou 11. Dans un format 6LoWPAN, TF = 01 implique un champ en ligne de 3 octets qui est transporté dans les champs non compressés. Ce champ en ligne est constitué de 2 bits d'ECN, suivis d'un bourrage de 2 bits, suivi de 20 bits d'étiquette de flux. Sachant que dans la présente norme, l'ECN est toujours activée et l'indication d'encombrement est transportée par la couche inférieure, l'ECN et le bourrage de 2 bits doivent être des 0 partout. Dans un format 6LoWPAN, TF = 11 implique que ce champ de 3 octets est élidé;
- NextHeaderNH = 1 pour indiquer que le prochain en-tête peut être inféré à partir du préfixe de l'en-tête de transport. Dans la présente norme, le prochain en-tête est toujours UDP;
- HLIM = HH. Ces bits sont utilisés par la présente norme pour indiquer le plan de compression de limite de saut tel que prévu dans 6LoWPAN;
- CID = 0 pour indiquer qu'aucune extension d'identificateur de contexte de 8 bits supplémentaire n'est utilisée;
- SAC = 1 pour spécifier la compression avec états pour l'adresse de source;

- SAM = 11 pour indiquer que tous les 128 bits de l'adresse source sont élidés (car, dans la présente norme, le pseudonyme D de 16 bits est transporté par la couche inférieure et est indexé dans la table ATT, qui fournit le contexte de conversion);
- M = 0 pour indiquer que l'adresse de destination n'est pas en multidiffusion;
- DAC = 1 pour spécifier la compression avec états pour l'adresse de destination;
- DAM = 11 pour indiquer que tous les 128 bits de l'adresse source sont élidés (car, dans la présente norme, le pseudonyme D de 16 bits est transporté par la couche inférieure et est indexé dans la table ATT, qui fournit le contexte de conversion).

10.5.4 Format (IPv6) d'en-tête complet

10.5.4.1 Usage prévu

Le format d'en-tête complet est utilisé principalement sur le réseau dorsal. Un appareil de terrain peut également utiliser le format d'en-tête complet au lieu du format de base ou du format activé par contrat, mais il convient de ne pas s'en servir.

Lorsque la NPDU atteint un routeur dorsal intermédiaire sur la DL, le routeur dorsal doit totalement étendre l'en-tête pour inclure tous les champs tels que définis dans l'en-tête IPv6. Les adresses de NL du tout premier appareil (initiateur) et de l'appareil de destination finale doivent être étendues à 128 bits pour désambiguïser leurs adresses DL16Addresses lors du routage à l'extérieur du sous-réseau D.

10.5.4.2 Format

Le Tableau 216 décrit le format d'en-tête IPv6.

Tableau 216 – Format d'en-tête de NL IPv6

<table>
<tr><th rowspan="2">Nombre d'octets</th><th colspan="8">bits</th></tr>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>1</td><td colspan="4">Version</td><td colspan="4">TrafficClass (bits 7..4)</td></tr>
<tr><td rowspan="3">3</td><td colspan="4">TrafficClass (bits 3..0)</td><td colspan="4">FlowLabel (bits 19..16)</td></tr>
<tr><td colspan="8">FlowLabel (bits 15..8)</td></tr>
<tr><td colspan="8">FlowLabel (bits 7..0)</td></tr>
<tr><td rowspan="2">2</td><td colspan="8">PayloadSize (bits 15..8)</td></tr>
<tr><td colspan="8">PayloadSize (bits 7..0)</td></tr>
<tr><td>1</td><td colspan="8">NextHeader</td></tr>
<tr><td>1</td><td colspan="8">HopLimit</td></tr>
<tr><td>16</td><td colspan="8">Adresse source</td></tr>
<tr><td>16</td><td colspan="8">Adresse de destination</td></tr>
<tr><td>(variable)</td><td colspan="8">Charge utile de réseau</td></tr>
</table>

Les champs comprennent:

- Version: le numéro de version IP de 4 bits doit être mis à 6;
- TrafficClass: les quatre bits de poids fort de ce champ de 8 bits doivent être utilisés pour transporter la priorité de 4 bits de la NPDU sur la dorsale. Le cinquième bit doit être 0 et le sixième bit doit être utilisé pour transporter le bit Discard Eligible (DE) de la NPDU sur la dorsale. Les deux bits de poids faible doivent transporter l'ECN;
- FlowLabel: la valeur de ce champ doit être telle que définie dans le format d'en-tête activé par contrat (voir 10.5.3.2). Ce champ doit être mis à zéro partout si l'ID de contrat n'est pas transporté comme partie intégrante du flux;

- PayloadSize: ce nombre entier non signé de 16 bits doit contenir la taille de la charge utile IPv6, c'est-à-dire, le reste de la NPDU suivant cet en-tête, en octets;
- NextHeader: ce champ de 8 bits doit être mis à 0x 0001 0001 (17 en décimale) identifiant l'en-tête suivant comme étant UDP;
- HopLimit: la valeur de ce champ de 8 bits doit être mise au nombre de sauts de NL (couche 3) nécessaires pour atteindre la destination. Si la NPDU est reçue sur un sous-réseau D avec l'en-tête de base, ce champ doit être mis à jour à la sortie du sous-réseau D par un routeur dorsal conformément à la table de routage du routeur. Ce champ est décrémenté de 1 par la NL de chaque appareil si la NL transmet la NPDU. La NPDU doit être rejetée si HopLimit est décrémenté à zéro;
- adresse source: ce champ doit contenir l'adresse N de 128 bits de l'initiateur de la NPDU.
- adresse de destination: ce champ doit contenir l'adresse IPv6Address de la destination finale (destinataire prévu) de la NPDU.

10.5.4.3 Relation à 6LoWPAN

Il convient de ne pas utiliser le format d'en-tête de NL complet dans une NPDU émise sur le sous-réseau D de la présente norme. Cependant, la norme n'exclut pas l'utilisation de l'en-tête complet sur des sous-réseaux D. Si elle est utilisée sur le sous-réseau D, la NPDU doit contenir le dispatch IPv6 conformément au Tableau 217.

Tableau 217 – En-tête de NL complet dans la DL

<table>
<tr><th rowspan="2">Nombre d'octets</th><th colspan="8">Bits</th></tr>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>1</td><td colspan="8">Dispatch IPv6</td></tr>
<tr><td rowspan="4">4</td><td colspan="4">Version</td><td colspan="4">TrafficClass (bits 7..4)</td></tr>
<tr><td colspan="4">TrafficClass (bits 3..0)</td><td colspan="4">FlowLabel (bits 19..16)</td></tr>
<tr><td colspan="8">FlowLabel (bits 15..8)</td></tr>
<tr><td colspan="8">FlowLabel (bits 7..0)</td></tr>
<tr><td>2</td><td colspan="8">PayloadSize (bits 15..8)</td></tr>
<tr><td></td><td colspan="8">PayloadSize (bits 7..0)</td></tr>
<tr><td>1</td><td colspan="8">NextHeader</td></tr>
<tr><td>1</td><td colspan="8">HopLimit</td></tr>
<tr><td>16</td><td colspan="8">Adresse source</td></tr>
<tr><td>16</td><td colspan="8">Adresse de destination</td></tr>
<tr><td>(variable)</td><td colspan="8">Charge utile de réseau</td></tr>
</table>

10.5.5 Format d'en-tête de fragmentation

10.5.5.1 Usage prévu

Si une NPDU avec une taille supérieure à dlmo.maxDSDUSize a besoin d'être émise sur la DL, la NPDU doit être fragmentée. Lorsqu'une NPDU doit être fragmentée, l'en-tête de fragmentation doit être inséré.

10.5.5.2 Format

L'en-tête de fragmentation contient un type de fragmentation de 5 bits, suivi d'un en-tête de 27 bits pour le premier fragment et d'un en-tête de 35 bits pour les fragments suivants. Les champs des en-têtes de base, activés par contrat ou complets, à partir de l'octet dispatch,

doivent être placés dans le premier fragment seulement. Le Tableau 218 montre le format d'en-tête de NL pour les NPDU fragmentées.

Tableau 218 – Format d'en-tête de NL pour les NPDU fragmentées

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
4..5	Type de fragmentation					En-tête de fragmentation		
	Autres champs d'en-tête de fragmentation (voir Tableau 219 et Tableau 220)							
(variable)	En-tête de base / activé par contrat / complet (pour le premier fragment seulement)							
(variable)	Charge utile de réseau							

Les champs comprennent:

- type de fragmentation: ce champ de 5 bits doit être mis à 0x1 1000 pour le premier fragment et à 0x1 1100 pour le deuxième fragment et les suivants;
- En-tête de fragmentation:
  - premier fragment: le premier fragment doit contenir le premier en-tête de fragment tel que défini dans le Tableau 219 et le texte suivant.

Tableau 219 – Format d'en-tête du premier fragment

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
2	Type de fragmentation					Datagram_size (bits 10..8)		
	1	1	0	0	0			
	Datagram_size (bits 7..0)							
2	Datagram_tag							

  - Fragments suivants: le deuxième segment et les suivants (jusqu'au dernier segment compris) doivent contenir un en-tête de fragmentation, tel que défini dans le Tableau 220 et le texte suivant.

Tableau 220 – Format d'en-têtes du second fragment et des fragments suivants

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
2	Type de fragmentation					Datagram_size (bits 10..8)		
	1	1	1	0	0			
	Datagram_size (bits 7..0)							
2	Datagram_tag							
1	Datagram_offset							

Tous les en-têtes de fragment comprennent:

- Datagram_size: ce champ de 11 bits code la taille de la NSDU entière avant fragmentation plus la taille de l'en-tête de base, activé par contrat ou complet. La valeur de Datagram_size doit être la même pour tous les fragments de la NSDU;
- Datagram_tag: il fournit l'identification pour la NLE de réassemblage. La NLE d'origine doit incrémenter Datagram_tag pour les NPDU fragmentées successives.

La valeur incrémentée de Datagram_tag est tronquée à gauche à 16 bits (à savoir, modulo 2^{16}) pour l'inclusion dans la NPDU. Pour réduire au maximum sa prévisibilité à un assaillant, la valeur initiale pour Datagram_tag doit être sélectionnée dans une distribution aléatoire uniforme. La valeur de Datagram_tag doit être la même pour tous les fragments d'une NSDU.

Les en-têtes du second fragment et des fragments suivants comprennent également:

– Datagram_offset: ce champ doit être présent seulement dans le deuxième fragment et les fragments suivants et il doit spécifier le décalage, en unités de 8 octets, du fragment par rapport au commencement de la charge utile de NPDU. (Le premier fragment de la NSDU a un décalage de zéro; la valeur implicite de Datagram_offset dans le premier fragment est zéro.) Ce champ a une longueur de 8 bits.

10.5.5.3 Relation à 6LoWPAN

Les formats d'en-tête de fragmentation dans la présente norme sont basés entièrement sur le format de l'IETF RFC 4944, sans changement. Comme dans l'IETF RFC 4944, le champ de Datagram_size est transporté dans tous les fragments, et le Datagram_offset est exprimé en unités de 8 octets. La fragmentation et le réassemblage peuvent se produire au niveau d'appareils intermédiaires et ne sont pas nécessairement de bout en bout.

11 Transport layer (couche transport)

11.1 Généralités

Le modèle de référence dans la présente norme est composé compose de cinq couches de protocoles. Dans ce modèle, le transport est la quatrième couche, immédiatement subordonnée à l'AL. Une TLE répond aux demandes de service d'une ALE supérieure en un TSAP et émet des demandes de service vers une NLE inférieure en un NSAP.

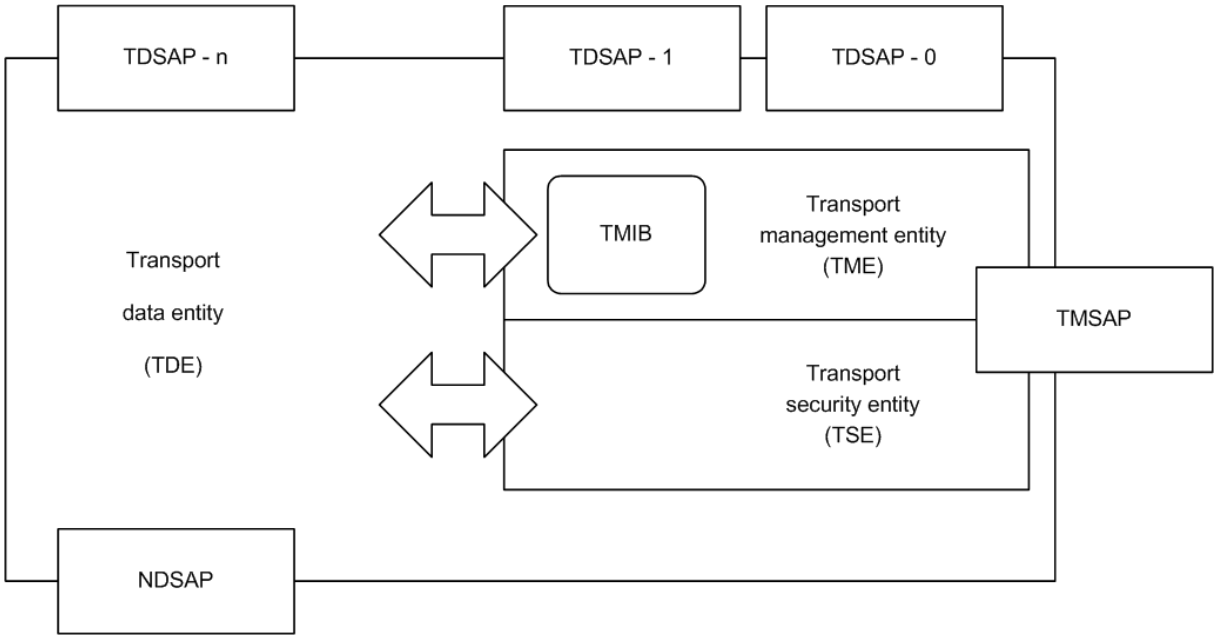
La TL est responsable de la communication de bout en bout et fonctionne seulement aux points d'extrémité de communication (par opposition aux appareils de routage).

Le TL fournit des services sans connexion, habituellement avec une sécurité fondée sur la cryptographie:

- le service sans connexion étend l'UDP sur l'IPv6, tel que défini dans l'IETF RFC 768 et l'IETF RFC 2460, en fournissant un contrôle d'intégrité de TDPU alternatif et plus sécurisé avec authentification cryptographique et, si configuré comme telle, avec la confidentialité;
- la sécurité est traitée d'une manière similaire à celle de la DL, mais de bout en bout plutôt que saut par saut;
- le service sans connexion étend également 6LoWPAN tel que défini par l'IETF RFC 6282. Lorsque la sécurité est activée, l'en-tête d'UDP peut être comprimé à un octet en élidant la somme de contrôle UDP et en s'appuyant sur le code d'intégrité des messages de transport (TMIC) qui est abouté et plus grand pour assurer l'intégrité de bout en bout.

11.2 Modèle de référence de TLE

Le modèle de référence de TLE d'état est représenté à la Figure 107.



Légende

Anglais	Français
Transport data entity (TDE)	Entité de données de transport (TDE)
Transport management entity (TME)	Entité de gestion de transport (TME)
Transport security entity (TSE)	Entité de sécurité de transport(TSE)

Figure 107 – Modèle de référence de TLE

La TLE inclut conceptuellement l'entité de gestion de transport (TME), l'entité de données de transport (TDE) et le composant de sécurité de transport (TSC).

La TDE a un TDSAP dédié pour le DMAP, à savoir TDE-0; un TDSAP qui est réservé pour SMAP, à savoir TDE-1; et un TDSAP pour chaque UAP, à savoir TDE-2 à TDE-n.

La TME configure et surveille les actions de la TLE. La base d'informations de gestion de transport (TMIB) (voir 11.6.2.5.2) maintient des informations abstraites pour la TME, qui sont accessibles au TMSAP comme partie intégrante du TLMO. Le TSC fournit les fonctions de sécurité de la TLE, basées sur les tables d'information de sécurité qui sont maintenues et surveillées par l'intermédiaire du TMSAP. La TDE utilise le TSC pour accomplir des opérations de sécurité sur les TPDU.

11.3 Entité de sécurité de transport

11.3.1 Généralités

Le TSC est conceptuellement un service compartimenté au sein de la TLE. La sécurité de TL peut protéger l'intégrité de l'unité de données de service de transport (TSDU) acheminée, de l'en-tête de transport et des adresses IPv6Addresses de points d'extrémité de transport. Si active, elle assure également la protection contre le retard excessif de TPDU et le rejet de TPDU, et peut chiffrer la TSDU pour la confidentialité.

11.3.2 Sécurisation de la TL

Le TSC et le DSC partagent la fonctionnalité. Le TSC est chargé de:

a) déterminer quel niveau de sécurité doit être appliqué à une session sécurisée en se basant sur les politiques;

b) à la réception d'une TPDU (comme tout ou partie d'une NSDU),

1) rejeter les TPDU non conformes;
2) rejeter les TPDUs qui échouent, selon la configuration d'association de T, soit
   i) au contrôle traditionnel d'intégrité d'UDP qui protège contre des erreurs non malveillantes, soit
   ii) aux contrôles d'authentification cryptographique qui protègent contre la modification tant accidentelle que délibérée de TPDU, contre le retard excessif de TPDU et le rejet de TPDU;
3) déchiffrer une TSDU protégée acheminée par la TPDU;

c) lors de la préparation d'une TPDU pour l'émission par l'intermédiaire d'une NLE,
   1) chiffrer les TSDU qui doivent avoir la protection de confidentialité pendant l'acheminement de TL; et
   2) selon la configuration de session, soit
      i) inclure la somme de contrôle d'intégrité d'UDP traditionnelle calculée pour protéger contre les erreurs non malveillantes, soit
      ii) inclure du support d'authentification cryptographique pour protéger contre la modification tant accidentelle que délibérée de TPDU, contre le retard excessif de TPDU et le rejet de TPDU.

La fonctionnalité du TSC est définie en 7.3.3.

Semblable à la sécurité de DL, la sécurité de TL utilise un chiffrement par blocs cryptographiques dans un mode de chiffrement par blocs de chiffrement authentifié générique, appelé le compteur avec CBC-MAC (CCM *), tel que défini par l'Annexe B de l'IEEE 802.15.4:2011. Le chiffrement par blocs par défaut est AES 128, mais d'autres chiffrements par blocs peuvent être utilisés lorsqu'exigés par la réglementation nationale.

CCM* permet l'authentification d'un message tout en chiffrant seulement une partie du message en question, laissant le reste du message (habituellement un en-tête) en texte clair. Lorsque cet appareil est activé pour une session particulière, la somme de contrôle d'UDP n'est pas utilisée et est élidée dans la forme comprimée de la TPDU, comme spécifié en 11.4.3.4. Elle est toujours présente dans la forme étendue de la TPDU, pour assurer la conformité à l'IETF RFC 2460 (UDP sur IPv6).

11.4 Entité de données de transport

11.4.1 Généralités

La TDE fournit des services sans connexion basés sur le protocole datagramme d'utilisateur (UDP, IETF RFC 768) sur IPv6 (IETF RFC 2460:1998), avec une compression telle que définie dans l'IETF RFC 6282:2011.

Les principales étapes de la construction de TPDU sont, dans l'ordre:

a) une TSDU est reçue en provenance d'une ALE locale par l'intermédiaire d'un TDSAP.

   NOTE Cette TSDU peut acheminer une seule APDU ou plusieurs APDU concaténées;

b) la TSDU est protégée et horodatée comme décrit en 7.3.3.2.1. La charge utile UDP résultante comprend un en-tête de sécurité de TL, la TSDU, et, lorsque la sécurité cryptographique est configurée, un TMIC. La TSDU est chiffrée pour la confidentialité si cela est ainsi indiqué par l'en-tête de sécurité de TL. La présence du TMIC, et sa taille, sont acheminées vers l'(les) appareil(s) de réception dans l'en-tête de sécurité de TL;

c) un en-tête d'UDP est ajouté en préfixe au début de la charge utile d'UDP. L'en-tête d'UDP peut être compressé. La compression implique d'élider la somme de contrôle UDP de l'en-tête d'UDP quand la charge utile d'UDP contient un autre contrôle d'intégrité. Un contrôle d'intégrité au sein de la charge utile d'UDP peut être un TMIC tel qu'indiqué dans l'en-tête de sécurité de TL et/ou un contrôle d'intégrité intégré dans une charge utile en tunnel.

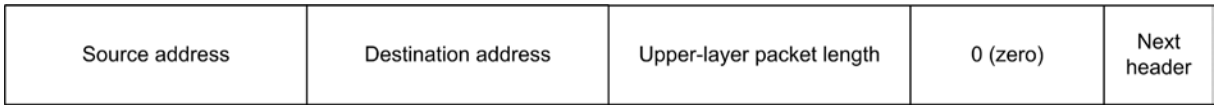
11.4.2 UDP sur IPv6

L'UDP (IETF RFC 768) fournit un mode sans connexion de communication d'ordinateurs dans l'environnement d'un ensemble interconnecté de réseaux d'ordinateurs.

L'UDP est essentiellement transparent au fonctionnement d'application, laissant à l'AL la commande et la responsabilité du fonctionnement correct du réseau. Des approches éprouvées, basées sur des normes, existent dans ce but. Les questions pertinentes sont documentées plus profondément en 11.4.4.

L'IETF RFC 768 suppose que l'IPv4 est utilisé comme protocole sous-jacent et définit le calcul d'une somme de contrôle d'UDP, utilisée pour la détection d'erreurs, qui couvre un pseudo-en-tête d'UDP d'information issu de l'en-tête de réseau IPv4, l'en-tête d'UDP, et la charge utile d'UDP. Si ce calcul donne un résultat de zéro, il est changé en 0xFFFF pour le placement dans l'en-tête d'UDP et la valeur de zéro est réservée pour indiquer qu'il n'y a aucun calcul de somme de contrôle.

L'IETF RFC 2460 spécifie les changements apportés à l'IETF RFC 768 pour adapter l'UDP à l'IPv6. Les changements apportés à l'UDP sont mineurs et se rapportent au calcul de la somme de contrôle d'UDP, qui devient obligatoire et inclut un pseudo-en-tête d'UDP modifié adapté pour l'IPv6, conformément à la Figure 108. Selon l'IETF RFC 2460, les récepteurs IPv6 rejettent les paquets UDP contenant une somme de contrôle de 0x0000 et consignent dans un journal cet événement comme étant une erreur.



Légende

Anglais	Français
Source address	Adresse source
Destination address	Adresse de destination
Upper-layer packet length	Longueur de paquet de couche supérieure
0 (zero)	0 (zéro)
Next header	Prochain en-tête

Figure 108 – Pseudo-en-tête UDP pour l'IPv6

Dans le pseudo-en-tête, la valeur dans le champ appelé "Next header" (prochain en-tête) est mise à 17 pour indiquer l'UDP et la valeur dans le champ appelé "Upper-layer packet length" (longueur de paquet de couche supérieure) est le même que le champ "Length" dans l'en-tête d'UDP et rend compte de la taille de la TPDU entière, y compris l'en-tête UDP.

Un pseudo-en-tête différent, soit le pseudo-en-tête TSS, est utilisé dans le traitement de la sécurité de TL, comme décrit en 7.3.3.2.1.

NOTE Voir l'IETF RFC 768, l'IETF RFC 2460, et les 4.5.2.1 et 7.3.3.2.1 pour comprendre les objectifs des présents pseudo-en-têtes.

11.4.3 Emission et compression d'en-tête UDP

11.4.3.1 Généralités

La TL prend en charge l'UDP non compressé pour l'émission et la réception d'une TPDU. Un appareil qui met en œuvre la TL de la présente norme doit prendre en charge l'UDP non compressé, mais il convient qu'il comprime l'en-tête UDP pour l'émission sur le réseau sans fil.

Le Tableau 221 décrit le codage d'en-tête UDP.

Tableau 221 – Codage d'en-tête UDP

Nombre d'octets	Bits (présentés dans la convention IEC, qui est différente de la convention IETF)						
	7	6	5	4	3	2	1..0
2	Source port: qui est en correspondance univoque (un à un) avec le TDAP de l'appareil de source						
2	Destination port: qui est en correspondance univoque (un à un) avec le TDAP de l'appareil de destination						
2	Size: la taille en octets de ce datagramme utilisateur, y compris cet en-tête et l'APDU						
2	Checksum: le complément à un de 16 bits de la somme de complément à un a) du pseudo-en-tête UDP de la Figure 108, dérivé de l'en-tête IP; b) de l'en-tête UDP, et c) de la TSDU éventuellement chiffrée, complétée d'octets zéro à la fin (s'il y a lieu) pour obtenir un multiple de deux octets. Au cours du calcul de la somme de contrôle, le champ "checksum" du TPDU de l'UDP est d'abord mis à zéro.						

La TL est également conforme à 6LoWPAN, qui spécifie une méthode pour comprimer l'UDP en utilisant un format Next Header Compression (LoWPAN_NHC) et oblige à utiliser la compression d'UDP sur le sous-réseau D local. Un appareil qui met en œuvre la TL de la présente norme doit ainsi prendre en charge toutes les combinaisons offertes par le LOWPAN_NHC pour la compression et l'extension de l'en-tête UDP.

La taille maximale d'une TSDU dépend d'une diversité de facteurs, tels que la capacité tampon à chaque TLE de point d'extrémité de session, le niveau sélectionné de sécurité de TL, et les caractéristiques de réseau. Le gestionnaire de système configure la taille maximale de TSDU sur une base de contrat pour rendre compte de ces considérations pertinentes et d'autres encore, par l'intermédiaire de la valeur de l'élément Assigned_Max_TSDU_Size, tel que décrit dans le Tableau 30.

11.4.3.2 Compression et restauration des numéros de port UDP

La compression pour les numéros de port d'UDP est décrite dans l'IETF RFC 6282:2011, 4.3. Il convient que ces numéros de port qui sont optimisés par le processus de compression soient assignés à des ports (AE) avec une plus grande fréquence d'utilisation. La compression optimale est obtenue lorsque les numéros de port tant source que destination commencent à un numéro de base P et sont exprimés comme étant P + short_port, où:

- P est le port de base, $61\,616_{10}$, à savoir, 0xF0B0;
- short_port est un nombre entier positif qui est $15 \leq$ (à savoir, 0x0F).

Dans un tel cas, la paire des ports de source et de destination est compressée en un seul octet qui a spécifié les valeurs de source et de destination comme short_ports, ce qui réduit la taille du champ de TPDU exigé pour acheminer des numéros de port UDP de quatre octets en un octet.

Lorsque les deux ports ne peuvent pas se trouver au sein de la plage 0xF0B0..0xF0BF, la taille de la TPDU peut encore être réduite d'un octet si le port source ou destination est dans la plage 0xF000..0xF0FF; dans ce cas l'octet de poids fort correspondant de 0xF0 est élidé. Il convient ainsi que les applications de serveur se mettent à l'écoute d'un port dans la plage 0xF000..0xF0FF. Par exemple, un appareil de terrain type a un DMAP à 0xF0B0 (codé comme short_port 0) et peut avoir son seul UAP à 0xF0B2 (codé comme short_port 2), car 0xF0B1 (codé comme short_port 1) est réservé pour le SMAP local.

11.4.3.3 Elision et restauration du champ UDP Size (taille d'UDP)

Si l'en-tête d'UDP n'est pas compressé, la taille de NSDU est égale à la taille de TPDU qui est placée dans l'en-tête d'UDP.

Si l'en-tête d'UDP est compressé, le champ UDP Size est toujours élidé en émission et il est inféré à la réception à partir de la taille de NSDU passée par la NLE de réception. Dans ce cas, la taille de TPDU est égale à la taille de NSDU plus la différence calculable entre les tailles d'un en-tête UDP complet et de l'en-tête UDP réel tel que compressé.

Même si aucun des autres champs dans l'en-tête d'UDP n'est compressé ou élidé, le fait de compresser le champ UDP Size réduit d'un octet la taille de la TPDU et permet l'utilisation de l'en-tête de base à la NL.

Par exemple, dans le cas optimal décrit dans le Tableau 223, l'en-tête UDP compressé requiert 2 octets, économisant ainsi 6 octets en comparaison avec l'en-tête UDP complètement étendu. La taille de TDPU pour l'en-tête UDP étendu est ainsi NSDU_Size + 6.

11.4.3.4 Elision et restauration de la somme de contrôle UDP

La TL est conforme aux règles 6LoWPAN et aux opérations définies dans l'IETF RFC 6282:2011, 4.3.2, comme suit:

- L'autorisation d'élider la somme de contrôle UDP pourrait venir de l'ALE ou TSC. Le TSC sous-entend la présence ou l'absence de cette somme de contrôle en spécifiant si le TMIC est présent ou non.
- Une ALE peut élider la somme de contrôle UDP dans les cas suivants:
  - tunnellisation: l'ALE source met en tunnel une PDU (de type non spécifié) qui possède son propre mécanisme d'intégrité qui assure au moins autant de protection que la somme de contrôle UDP de 16 bits;
  - MIC d'application: l'ALE source applique un contrôle d'intégrité de message de bout en bout d'au moins 16 bits (par exemple, comme partie intégrante d'un échange de clé).
- Si la NLE locale est un routeur dorsal, alors le routeur doit recalculer la somme de contrôle élidée en se basant sur le paquet reçu comme spécifié dans l'IETF RFC 2460 et doit placer le résultat de ce calcul dans l'en-tête UDP reformé avant émission sur un réseau dorsal IPv6.
- Si la NLE est la destination du paquet et a connaissance de la présence du TMIC dans la TPDU, elle peut complètement omettre l'opération de somme de contrôle UDP (ni recalculer ni vérifier la somme de contrôle).
- Un routeur dorsal qui transmet un paquet IPv6 dans le sous-réseau D utilise le champ contrôle de sécurité dans l'en-tête de sécurité de TPDU pour déterminer si un TMIC est présent ou pas. Lors de l'accomplissement de la compression UDP, il convient que le routeur dorsal élide la somme de contrôle UDP si le TMIC est présent, mais il ne doit pas élider la somme de contrôle UDP si le TMIC n'est pas présent. Réciproquement, le fait que la somme de contrôle ne soit pas compressée n'est pas une indication qu'il n'y a pas de TMIC dans la TPDU.

6LoWPAN permet la compression de somme de contrôle UDP seulement lorsque la protection alternative au sein de la TPDU assure une protection d'intégrité au moins aussi grande que la somme de contrôle UDP, y compris la protection de bout en bout pour tous champs que la somme de contrôle UDP protégerait. Pour satisfaire à cette exigence, la présente norme définit un pseudo-en-tête TSC qui est inclus dans le calcul de TMIC, comme décrit en 7.3.3.2.1. Comparé au pseudo-en-tête UDP normalisé selon l'IETF RFC 2460, le pseudo-en-tête de TSC inclut les ports UDP, mais n'inclut pas la taille de charge utile, car cela est implicitement protégé par le TMIC. La structure du pseudo-en-tête de TSC est définie au Tableau 47.

Dans le cas d'une TPDU préparée pour l'émission, les adresses IPv6Addresses source et destination sont obtenues à partir des informations de contrat; l'ID de contrat lui-même est passé par l'ALE comme un paramètre associé à la TSDU. Le champ Next header du pseudo-en-tête est mis à 17 (UDP). Le port source est obtenu à partir du contexte de TL associé au TSAP, alors que le port de destination est un autre paramètre associé à la TSDU. Le pseudo-en-tête de TSC a été modélisé comme s'il était construit par la TDE et passé avec la TSDU au TSC pour le traitement de sécurité.

Le TSC utilise le pseudo-en-tête de TSC dans le calcul du TMIC en l'ajoutant en préfixe au début de la TSDU, mais le pseudo-en-tête de TSC n'est pas chiffré quand le chiffrement doit être accompli sur la TSDU. Une fois que le processus cryptographique est achevé, le TSC doit enlever de la TSDU traitée le pseudo-en-tête de TSS, ajouter ses propres en-têtes et en-queues, et retourner à la TDE la charge utile UDP protégée. Le TDE doit compléter la TPDU en ajoutant en préfixe, et habituellement en compressant l'en-tête UDP pour former la NSDU qui est passée à la NL.

Dans le cas d'une TPDU reçue, les adresses IPv6Addresses source et destination et les bits de priorité sont passés par la NL comme paramètres associés à la NSDU, et les ports sont obtenus à partir de l'en-tête UDP dans la TPDU une fois que l'en-tête UDP a été étendu. Le TDE doit remplir le pseudo-en-tête, retirer de la NSDU l'en-tête UDP et passer à la TSC la charge utile UDP protégée obtenue accompagnée du pseudo-en-tête et des bits de priorité pour le traitement de sécurité.

Sachant qu'une NPDU pourrait être reçue hors séquence, peut-être en raison de ses valeurs de réglage de sécurité (voir Tableau 205), le TSC peut utiliser ces informations relatives à la priorité pour partitionner son cache parallèle, optimisant potentiellement sa capacité, dans le cadre des ressources mémoire limitées, à valider des TPDU qui avaient encouru un retard de transit substantiel. Les informations relatives à la priorité sont acheminées avec la NSDU dans la primitive N-DATA.indication reçue par la TLE en provenance d'une NLE locale; la TLE transmet ces informations et la TPDU au TSC.

Le TSC de réception enlève de la TSDU protégée les en-têtes et les en-queues de sécurité de TL, ajoute en préfixe le pseudo-en-tête de TSC, les ports UDP, et les en-têtes de sécurité (voir Figure 41), et ensuite accomplit les vérifications de sécurité. Si un contrôle d'intégrité est appliqué, la TLE peut vérifier si des parties critiques des informations de la NL et de la TL ont été modifiées pendant le transport de bout en bout.

Si le champ somme de contrôle de couche supérieure UDP est élidé, il appartient à l'appareil de réception sur le sous-réseau D sans fil de recalculer la somme de contrôle UDP comme partie intégrante du processus d'inversion de la compression 6LoWPAN, s'il y a lieu, avant la transmission plus avant de la NPDU d'acheminement vers une destination à l'extérieur du sous-réseau D sans fil. Cette étape d'inversion de compression est nécessaire en particulier pour un appareil intermédiaire, tel qu'un routeur de frontière de BBR, qui transmet la NPDU non compressée sur un réseau différent.

NOTE Il est inutile pour la TLE de destination adressée de reconstituer ainsi la TPDU non compressée.

Si la somme de contrôle UDP n'est pas élidée, alors la somme de contrôle UDP doit être calculée en émission après que l'opération de sécurité a été complète et les champs de sécurité ont été préréglés pour le calcul. La charge utile UDP qui est utilisée pour la somme de contrôle UDP doit inclure toutes les données depuis l'en-tête d'UDP jusqu'à la fin de la TPDU, y compris les champs de sécurité, les données d'application, et les champs de MIC, indépendamment du fait que les données d'application soient chiffrées ou pas. En réception, elle doit être vérifiée par la TDE avant l'opération de TSC, à moins que la TDE ne sache que le TSC vérifie le MIC, auquel cas la somme de contrôle peut être ignorée.

11.4.4 TSAP et ports UDP

L'appareil autoconfigure une adresse IPv6Address locale à une liaison IPv6 en se basant sur son adresse EUI64Address. Une fois que le processus de rattachement au sous-réseau est

achevé, un appareil possède également au moins une adresse IPv6Address globale. La présente norme autorise, mais n'exige pas, qu'un appareil prenne en charge plus d'une adresse IPv6Address globale. Pour une adresse IPv6Address donnée, un port UDP doit être associé avec un UAP maximum.

Le port est utilisé comme le port local pour toutes les émissions depuis/vers l'UAP en question en utilisant cette adresse IPv6Address. Un multiplexage plus poussé entre les objets UAP est de la responsabilité de l'ALE, se fait au sein des TSDU et est donc transparent à la TLE. En conséquence, seul un nombre limité de ports est réellement utilisé dans le système, et il n'y a aucune allocation dynamique de port après la phase d'initialisation (ou de réinitialisation) des applications.

Chaque UAP doit être responsable de connaître son numéro de port UDP et de l'enregistrer auprès de la TLE. A cet instant, un TLSAP doit être mis en correspondance sur une base univoque (1:1) avec l'UAP et le port local pour l'adresse IPv6Address. Une adresse de processus d'application doit être constituée d'une adresse d'appareil concaténée avec son identificateur de TLSAP.

11.4.5 Bonne citoyenneté de réseau

Les SDU (N) sont échangées avec les couches supérieures et inférieures par l'intermédiaire de primitives DATA (N). Une mise en œuvre peut rapporter l'encombrement transitoire dans une couche inférieure par l'intermédiaire d'un code d'achèvement spécifique dans une primitive (N)-DATA.confirm qui est acheminée en montant tout le chemin vers la suite de protocoles. A la réception d'un code d'achèvement indiquant un encombrement transitoire, il convient qu'une ALE diffère la réémission de la SDU (N) d'une quantité de repli exponentiel ou abandonne simplement la SDU (N).

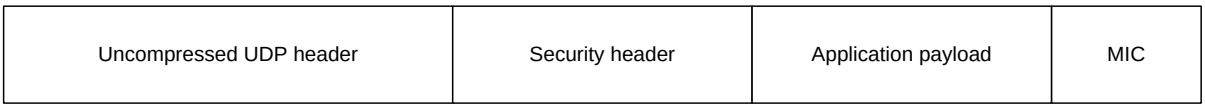
Sachant que la TL est basée sur l'UDP, il est souhaitable que les UAP prennent en charge le contrôle de flux compatible TCP ou le contrôle de débit compatible TCP (TFRC, IETF RFC 5348), en fonction de la nature du flux, afin de protéger les futures utilisations du réseau et partager équitablement le réseau dorsal.

Cela est habituellement réalisé en mettant en œuvre un protocole de blocage au niveau de l'AL qui applique un temps de propagation aller-retour (RTT) entre les TPDU ou utilise le RTT pour calculer un temporisateur de récupération sur perte. Dans le cas des échantillons périodiques, cela peut être réalisé en réservant la largeur de bande ou en gardant la période d'échantillonnage au-dessus de la valeur initiale de RTT pour TCP de 3 s (voir IETF RFC 6298). Des fréquences d'échantillonnage plus élevées peuvent être utilisées dans certaines applications.

11.5 Codage de TPDU

11.5.1 Généralités

Une TPDU échangée entre les deux TLE d'une communication doit être composée d'un en-tête UDP sous sa forme non compressée, d'un en-tête d'informations de sécurité, de la TSDU, et d'un TMIC, conformément à la Figure 109.



Légende

Anglais	Français
Uncompressed UDP header	En-tête UDP décompressé
Security header	En-tête de sécurité
Application payload	Charge utile d'application

Figure 109 – Structure de TPDU

Les données de NSDU passées par la TLE expéditrice à une NLE locale par l'intermédiaire d'un NSAP incluent toute compression d'en-tête UDP qui a été appliquée à la TPDU. Si la compression a été appliquée, l'octet 6LoWPAN _NHC-for-UDP est le premier octet de la NSDU.

La paire d'adresses IPv6Addresses, le type de transport (UDP=17) et la taille de la NSDU ne sont pas présents dans la NSDU, aussi sont-ils passés comme paramètres associés. Le fait de savoir si, oui ou non, la compression 6LoWPAN_NHC a eu lieu est également passé comme paramètre.

11.5.2 Compression d'en-tête – Codage de protocole datagramme d'utilisateur

Lorsque l'en-tête UDP n'est pas compressé, la NSDU est identique à la TPDU et l'en-tête UDP doit être présent au complet sans un octet de codage 6LoWPAN_NHC-for-UDP.

Si l'un quelconque des champs d'en-tête UDP est compressé, un octet de codage 6LoWPAN_NHC-for-UDP doit être ajouté en préfixe afin de décrire la compression. La NSDU doit être formée en remplaçant l'en-tête d'UDP au début de la TPDU par l'octet de codage 6LoWPAN_NHC-for-UDP suivi des données compressées. L'octet de codage 6LoWPAN_NHC-for-UDP est structuré conformément au Tableau 222.

Tableau 222 – Octet de codage 6LoWPAN_NHC-for-UDP

Nombre d'octets	Bits (présentés dans la convention IEC, qui est différente de la convention IETF)							
	7	6	5	4	3	2	1	0
1	1	1	1	1	0	C	P	

Les champs C (compression de somme de contrôle) et P (compression de ports) sont définis comme suit:

Champ C (1 bit):

0: La somme de contrôle de 16 bits de la TPDU est acheminée dans la TPDU;

1: La somme de contrôle de 16 bits de la TPDU est élidée de la TPDU.

Champ P (2 bits):

00: la TPDU achemine les ports source et destination sous forme de valeurs de 16 bits;

01: la TPDU achemine le port source sous forme d'une valeur de 16 bits alors que seuls les 8 bits de poids faible du port destination, qui se situe dans la plage 0xF000..0xF0FF, sont acheminés dans la TPDU;

10: la TPDU achemine le port destination sous forme d'une valeur de 16 bits alors que seuls les 8 bits de poids faible du port source, qui se situe dans la plage 0xF000..0xF0FF, sont acheminés dans la TPDU;

11: TPDU achemine seulement les 4 bits de poids faible des ports source et destination, qui se situent dans la plage 0xF0B0..0xF0BF.

Les champs compressés apparaissent dans le même ordre qu'ils le font dans le format d'en-tête UDP spécifié dans l'IETF RFC 768.

Lorsque le plus haut degré de compression est réalisé, seuls les numéros de port short_port compressés sont transportés après l'octet de codage 6LoWPAN_NHC-for-UDP conformément au Tableau 223.

Tableau 223 –Codage optimal d'en-tête UDP

Nombre d'octets	Bits (présentés dans la convention IEC, qui est différente de la convention IETF)						
	7	6	5	4	3	2	1..0
1	1	1	1	1	0	1 (la somme de contrôle est élidée)	11 (les ports source et destination sont compressés)
2	source short_port				destination short_port		

Les valeurs étendues pour le port source et les ports de destination compressés doivent être calculées en utilisant la formule:

UDP_port_number = P + short_port

où short_port est le numéro de port compressé de 4 bits et P est la valeur 0xF0B0 (61 616_{10}). En d'autres termes, short_port achemine les 4 bits de poids faible du numéro de port UDP.

Au moment du processus de rattachement au sous-réseau, l'appareil de terrain n'a pas le support cryptographique pertinent pour calculer un TMIC. Sachant que cette situation exige que le TMIC soit omis, l'appareil doit calculer la somme de contrôle UDP telle qu'exigée par l'IETF RFC 768 et l'IETF RFC 2460 et doit utiliser les adresses IPv6Addresses locales à une liaison pour calculer et envoyer la somme de contrôle et émettre. Dans ce cas, si les ports UDP peuvent être compressés, l'en-tête UDP codé est formaté tel que présenté dans le Tableau 224.

Tableau 224 – Codage d'en-tête UDP avec somme de contrôle et numéros de port compressés

Nombre d'octets	Bits						
	7	6	5	4	3	2	1..0
1	1	1	1	1	0	0 (la somme de contrôle est présente)	11 (les ports source et destination sont compressés)
2	source short_port				destination short_port		
3	Somme de contrôle UDP						
4							

11.5.3 En-tête de sécurité de TPDU

Pour des informations relatives à l'en-tête de sécurité d'une TPDU, le codage et le décodage, voir 7.3.3.6.

11.6 Modèle de TL

11.6.1 Généralités

Une TLE fournit deux interfaces:

- un TDSAP de TDE pour chaque UAP et un pour le DMAP;
- un TMSAP de TME pour le DMAP.

Ces interfaces sont illustrées dans le modèle de référence de protocole de la présente norme, montré à la Figure 16.

Toutes les interfaces entre la TLE et les entités de couche ou entités de gestion adjacentes sont des interfaces internes au sein de l'appareil, et sont donc inobservables. Par conséquent, elles ne sont pas soumises à une normalisation.

Les couches supérieures utilisent le TDSAP pour échanger des données communiquées par l'intermédiaire de la TLE. Il y a une instance de TDSAP distincte pour chaque UAP, plus une instance pour le DMAP.

Le TDSAP inclut une fonction de multiplexeur qui adapte l'espace de noms d'adresse de l'ALE à l'espace natif de noms d'adresse de la TLE. Le DMAP utilise le TMSAP pour configurer la TLE et surveiller son fonctionnement.

La TLE communique avec une NLE locale en utilisant un NDSAP.

11.6.2 Services de données

11.6.2.1 Généralités

Pour des besoins d'illustration, un exemple de jeu de primitives est fourni dans la présente norme. La TL offre un service non connecté basé sur le modèle UDP.

Une TLE utilise l'interface fournie par le TDSAP pour émettre et recevoir des unités de données de protocole avec l'ALE.

Le TDSAP transfère la TSDU, avec des paramètres d'information de commande et de statut.

11.6.2.2 T-DATA.request

11.6.2.2.1 Généralités

La primitive T-DATA.request ordonne à la TL d'émettre une unité de données de protocole.

11.6.2.2.2 Sémantique de la primitive de service

La sémantique de la primitive T-DATA.request se présente comme suit:

```
T-DATA.request   (
                 ContractId
                 TSDU_size
                 TSDU
                 Priority
                 DE
                 TDSAP
                 \Destination_Port)
```

Le Tableau 225 spécifie les éléments pour la primitive T-DATA.request.

Tableau 225 – Eléments pour la primitive T-DATA.request

Nom de l'élément	Identificateur de l'élément	Type scalaire de l'élément
ContractId (identifie les ressources TL contractuelles associées à l'unité de données de protocole)	1	Unsigned16 Valeurs nommées: 0: aucun contrat
TSDU_size (taille en octets de l'unité de données de protocole passée à partir de l'ASL)	2	Unsigned16
TSDU (la TSDU devant être émise)	3	OctetString
N-priority (identifie la priorité au sein de la NL de cette TSDU)	4	Unsigned2
DE (identifie le Discard Eligibility de cette TSDU)	5	Unsigned1
TDSAP (ID du TDSAP qui accorde le service UDP sur SourcePort)	6	Unsigned16
Destination_Port [port (distant) de destination UDP pour la TSDU]	7	Unsigned16

La TLE maintient une table indexée par TDSAP qui contient (implicitement ou explicitement) l'adresse IPv6Address locale et (explicitement) le port local pour l'émission. Cette table est accessible pour obtenir l'adresse IPv6Address source et le port pour l'émission sur un TDSAP donné. L'adresse IPv6Address de destination est obtenue à partir du champ Destination_Address dans les informations de contrat, indexé par l'ID de contrat.

La TLE ne retient pas les informations d'état relatives au port distant; ainsi, les informations qui sont passées par l'UAP sont passées dans le TPDU sans vérification.

Le paramètre N-priority communique la priorité N qui doit être utilisée par la NL. Les valeurs de réglage de N-priority ne sont pas utilisées dans la TL, mais elles sont passées par la NL à la DL, où elles sont utilisées pour choisir la classe de priorité dans l'en-tête de DL.

11.6.2.2.3 Utilisation appropriée

Une ALE invoque la primitive T-DATA.request pour passer une TSDU à une TLE locale pour émission sur le réseau.

11.6.2.2.4 Effet à la réception

A la réception de la primitive T-DATA.request, la TLE recherche le niveau de sécurité T dans le KeyDescriptor pour déterminer le traitement exigé par le TSC local, construit l'en-tête de TPDU, forme la TPDU, et génère la primitive N-DATA.request pour une NLE locale au niveau d'un NSAP.

11.6.2.3 T-DATA.confirm

11.6.2.3.1 Généralités

La primitive T-DATA.confirm est utilisée par la TLE pour répondre à une primitive T-DATA.request. La confirmation est immédiate et dit à l'ALE demandeuse soit que la demande a été couronnée de succès, soit qu'une erreur a été détectée. Les conditions d'erreur incluent des questions telles qu'un ID de contrat non reconnu, une taille de TSDU qui est incorrecte ou excessive, ou une erreur transitoire interne telle qu'un encombrement de réseau au-delà de la capacité du TSC et de ses agents pour les traiter.

11.6.2.3.2 Sémantique de la primitive de service

La sémantique de T-DATA.confirm se présente comme suit:

T-DATA.confirm (
ContractId
TDSAP
status
)

Le Tableau 226 spécifie les éléments pour T-DATA.confirm.

Tableau 226 – Eléments pour la primitive T-DATA.confirm

Nom de l'élément	Identificateur de l'élément	Type scalaire de l'élément
ContractId (identifie les ressources TL contractuelles associées à la TPDU)	1	Type: Unsigned16 Valeurs nommées: 0: aucun contrat
TDSAP (ID du TDSAP qui accorde le service UDP sur SourcePort)	2	Type: Unsigned16
Status (le résultat de la primitive T-DATA.request)	3	Type: Unsigned Plage valide: (voir Tableau 227)

Le Tableau 227 fournit les codes de statut pour la primitive T-DATA.confirm.

Tableau 227 – Codes de statut de la primitive T-Data.confirm

Nom	Description
SUCCESS	TSDU acceptée
TRANSIENT_FAILURE	La TSDU est rejetée, mais peut être tentée de nouveau après une brève période de temps
FAILURE	Echec générique: TSDU rejetée sans raison explicite
INVALID_CONTRACT	Echec spécifique: ID de contrat non reconnu; TSDU rejetée
INVALID_LENGTH	Echec spécifique: La TSDU est plus grande que l'Assigned_Max_TSDU_Size; rejetée
PORT_ERROR	Echec spécifique: SourcePort n'est pas enregistré pour le TDSAP; TSDU rejetée
SAP_ERROR	Echec spécifique: TDSAP inconnu; TSDU rejetée

11.6.2.3.3 Moment de sa génération

Une TLE génère une primitive T-DATA.confirm en réponse à une primitive T-DATA.request locale. La primitive T-DATA.confirm retourne de façon synchrone soit un statut de succès, soit un code d'erreur approprié.

11.6.2.3.4 Utilisation appropriée

La primitive T-DATA.confirm notifie à l'ALE le résultat de sa demande d'émettre une TSDU.

11.6.2.4 T-DATA.indication

11.6.2.4.1 Généralités

La primitive T-DATA.indication est utilisée pour transférer une TSDU vers l'ALE. Elle est générée lorsqu'une TPDU a été reçue avec succès en provenance d'une NLE locale et a été traitée par la TLE.

Une TPDU reçue qui échoue au traitement de sécurité T ou qui spécifie un port de destination non enregistré est rejetée à la réception. De telles erreurs sont journalisées dans la PIB de la TLE.

11.6.2.4.2 Sémantique de la primitive de service

La sémantique de la primitive T-DATA.indication se présente comme suit:

```
T-DATA.indication (
                SrcAddr
                SrcPort
                TSDU_size
                TSDU
                ECN
                TDSAP
                transportTime
                )
```

Le Tableau 228 spécifie les éléments pour la primitive T-DATA.indication.

Tableau 228 – Eléments pour la primitive T-Data.indication.

Nom de l'élément	Identificateur de l'élément	Type scalaire de l'élément
SourceNetworkAddress (adresse IP de l'extrémité distante)	1	Type: IPv6Address
SourcePort (port source UDP dans la TPDU entrante)	2	Type: Unsigned16
TSDU_size (taille en octets dans la TSDU d'accompagnement)	3	Type: Unsigned16
TSDU (le contenu reçu de couche supérieure de la TPDU)	4	Type: OctetStringN
ECN (bits de notification d'encombrement explicite)	5	Type: Unsigned2
TDSAP (ID du TDSAP qui accorde le service UDP et concorde avec un port local)	6	Type: Unsigned8
TransportTime (temps de transit de bout en bout du TSC d'origine au TSC de réception)	7	Type:Unsigned16
NOTE TransportTime est relatif à tpduMaxAge.		

Une TLE maintient une table qui contient le TDSAP, qui est indexée (implicitement ou explicitement) par l'adresse locale et (explicitement) par le port local pour la réception. La table en question est accessible pour trouver le TDSAP utilisé pour passer la TSDU à l'UAP.

11.6.2.4.3 Utilisation appropriée

Une TLE invoque la primitive T-DATA.indication pour notifier à l'ALE adressée une TSDU reçue.

11.6.2.4.4 Effet à la réception

A la réception de la primitive T-DATA.indication, l'ALE traite la TSDU reçue.

11.6.2.5 Services de gestion

11.6.2.5.1 Généralités

Le service de la gestion de la TLE est commandé par l'objet de gestion de TL (TLMO) dans le DMAP. Le TLMO commande les fonctionnalités de la TLE en:

- mesurant la latence de la TL et en réalisant les adaptations connexes pour se conformer dynamiquement aux exigences de latence; et
- en recueillant des paramètres opérationnels.

Le TLMO utilise son interface de TMSAP pour configurer et commander le fonctionnement de la TLE. La TME de la TLE fournit une TMIB qui maintient les informations d'état nécessaires pour mettre en œuvre la fonctionnalité de TMSAP.

11.6.2.5.2 Attributs

Le Tableau 229 spécifie les attributs du TLMO.

Tableau 229 – Attributs de TLMO *(1 de 2)*

Nom du type d'objet normalisé: TL management object (TLMO, objet de gestion de TL)				
Identificateur du type d'objet normalisé: 122				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Reserved:	1	Réservé par la norme	-	-
MaxNbOfPorts	2	Nombre de ports actifs	Type: Unsigned8 Classification: Constant Accessibilité: Lecture seule Valeur par défaut: Dépendant de l'appareil Plage valide: 2..255	La valeur minimale couvre un appareil type avec un seul DMAP et un seul TDSAP UAP
TPDUin	3	Compteur rapportant le nombre de TPDU reçues	Type: Unsigned32 Classification: Dynamic Accessibilité: Lecture seule Valeur par défaut: 0	Incrémenté avec chaque TPDU reçue en provenance d'une TLE distante
TPDUinRejected	4	Compteur rapportant le nombre de TPDU rejetées	Type: Unsigned32 Classification: Dynamic Accessibilité: Lecture seule Valeur par défaut: 0	Incrémenté avec chaque unité de données reçue en provenance d'une TLE distante qui avait été rejetée (pour des raisons de sécurité, par exemple). Noter qu'il n'existe pas de tel compteur au sein du DSMO

Tableau 229 *(2 de 2)*

Nom du type d'objet normalisé: TL management object (TLMO, objet de gestion de TL)				
Identificateur du type d'objet normalisé: 122				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
TSDUout	5	Compteur rapportant le nombre de TSDU passées à l'ALE locale	Type: Unsigned32 Classification: Dynamic Accessibilité: Lecture seule Valeur par défaut: 0	Incrémenté avec chaque TPDU reçue en provenance d'une TLE distante qui a entraîné l'acheminement d'une TSDU contenue vers une ALE locale
TSDUin	6	Compteur rapportant le nombre de TSDU reçues	Type: Unsigned32 Classification: Dynamic Accessibilité: Lecture seule Valeur par défaut: 0	Incrémenté avec chaque TSDU reçue en provenance d'une ALE locale
TSDUinRejected	7	Compteur rapportant le nombre de TSDU rejetées	Type: Unsigned32 Classification: Dynamic Accessibilité: Lecture seule Valeur par défaut: 0	Incrémenté avec chaque TSDU reçue en provenance d'une ALE locale qui est rejetée
TPDUout	8	Compteur rapportant le nombre de TPDU passées à la NL	Type: Unsigned32 Classification: Dynamic Accessibilité: Lecture seule Valeur par défaut: 0	Incrémenté avec chaque TSDU reçue en provenance d'une ALE locale qui est acheminée et acceptée par une NLE locale
IllegalUseOfPortAlertDescriptor	9	Utilisé pour changer la priorité d'alerte IllegalUseOfPort; cette alerte peut également être activée ou désactivée	Type: Alert report descriptor Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: [true, 8] -- medium	-
TPDUonUnregisteredPortAlertDescriptor	10	Utilisé pour changer la priorité d'alerte TPDUonUnregisteredPort ; cette alerte peut également être activée ou désactivée	Type: Alert report descriptor Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: [true, 4] -- low	-

Pour chaque attribut, le TLMO fournit des méthodes lecture et écriture disponibles au DMAP. Ces méthodes sont mises en œuvre en demandant des services TME au travers du TMSAP.

11.6.2.5.3 Méthodes

En plus des services lecture et écriture pour les attributs, des méthodes de TLMO complémentaires fournissent l'accès à des services de TME au travers du TMSAP.

Le Tableau 230 décrit la méthode "Reset".

Tableau 230 – Méthodes de l'objet de gestion de TL – Reset

<table>
<tr><th colspan="5">Nom du type d'objet normalisé: TL management object (TLMO, objet de gestion de TL)</th></tr>
<tr><th colspan="5">Identificateur du type d'objet normalisé: 122</th></tr>
<tr><th>Nom de méthode</th><th>ID de méthode</th><th colspan="3">Description de la méthode</th></tr>
<tr><td rowspan="8">Reset</td><td>1</td><td colspan="3">Réinitialise le transport aux états initiaux</td></tr>
<tr><td colspan="4">Arguments d'entrée</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td>1</td><td>Forced</td><td>Boolean</td><td>‘Forced’ signifie que toutes les données sont mises à jour sans la moindre interaction avec d'autres entités. Les tables relatives au TSAP sont vidées, la mémoire connexe est libérée et tous les compteurs sont mis à 0.</td></tr>
<tr><td colspan="4">Arguments de sortie</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td>1</td><td>Status</td><td>Unsigned8</td><td>0 = réussite, > 0 échec</td></tr>
</table>

Le Tableau 231 décrit la méthode Halt.

Tableau 231 – Méthodes de l'objet de gestion de TL – Halt

Nom du type d'objet normalisé: TL management object (TLMO, objet de gestion de TL)				
Identificateur du type d'objet normalisé: 122				
Nom de méthode	**ID de méthode**	**Description de la méthode**		
Halt	2	Arrête un port et le remet à son état initial. Similaire à une réinitialisation, mais sa portée est ramenée à un seul port UDP. Tout le trafic est interrompu sur le port en question et le TSAP a besoin d'être rouvert pour le port en question pour redevenir opérationnel. Les entrées de table relatives au TSAP pour le port sont vidées et la mémoire connexe est libérée		
	Arguments d'entrée			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	DeviceAddress	IPv6Address	L'adresse IPv6Address de cet appareil
	2	PortNumber	Unsigned16	Le port à arrêter
	Arguments de sortie			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	Status	Unsigned8	Valeurs nommées: 0: réussite, 1: échec générique, 2: mauvais numéro de port

Le Tableau 232 décrit la méthode PortRangeInfo.

Tableau 232 – Méthodes de l'objet de gestion de TL – PortRangeInfo

Nom du type d'objet normalisé: TL management object (TLMO, objet de gestion de TL)				
Identificateur du type d'objet normalisé: 122				
Nom de méthode	**ID de méthode**	**Description de la méthode**		
PortRangeInfo	3	Rapporte les informations relatives à la plage de ports UDP		
	Arguments d'entrée			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	DeviceAddress	IPv6Address	L'adresse IPv6Address de cet appareil
	Arguments de sortie			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	Status	Unsigned8	0 = réussite, > 0 échec
	2	NbActivePorts	Unsigned16	Nombre de ports actifs
	3	FirstActivePort	Unsigned16	Premier port actif
	4	LastActivePort	Unsigned16	Dernier port actif

Le Tableau 233 décrit la méthode GetPortInfo.

Tableau 233 – Méthodes de l'objet de gestion de TL – GetPortInfo

Nom du type d'objet normalisé: TL management object (TLMO, objet de gestion de TL)				
Identificateur du type d'objet normalisé: 122				
Nom de méthode	**ID de méthode**	**Description de la méthode**		
GetPortInfo	4	Rapporte les informations relatives au port UDP pour un port UDP donné ou le premier port UDP actif		
	Arguments d'entrée			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	DeviceAddress	IPv6Address	L'adresse IPv6Address de cet appareil
	2	PortNumber	Unsigned16	Le port dont les infos sont demandées (0 = le plus bas)
	Arguments de sortie			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	Status	Unsigned8	0 = réussite, > 0 échec
	2	PortNumber	Unsigned16	Ce numéro de port
	3	UID	Unsigned32	ID d'application du propriétaire
	4	Compressed	Boolean	La compression s'applique à ce port
	5	TPDUsInOK	Unsigned32	Nombre de TPDU acceptées sur le port
	6	TPDUsInKO	Unsigned32	Nombre de TPDU rejetées sur le port
	7	TPDUsOutOK	Unsigned32	Nombre de TPDU envoyées sur le port
	8	TPDUsOutKO	Unsigned32	Nombre d'échecs d'émission de TPDU

Le Tableau 234 décrit la méthode GetNextPortInfo.

Tableau 234 – Méthodes de l'objet de gestion de TL – GetNextPortInfo

Nom du type d'objet normalisé: TL management object (TLMO, objet de gestion de TL)				
Identificateur du type d'objet normalisé: 122				
Nom de méthode	**ID de méthode**	**Description de la méthode**		
GetNextPortInfo	5	Rapporte les informations relatives au port UDP pour un port UDP donné ou le premier port UDP actif		
	Arguments d'entrée			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	DeviceAddress	IPv6Address	L'adresse IPv6Address de cet appareil
	2	PortNumber	Unsigned16	Le précédent port à partir duquel des infos sont demandées
	Arguments de sortie			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	Status	Unsigned8	0 = réussite, > 0 échec
	2	PortNumber	Unsigned16	Le port pour lequel des infos sont rapportées
	3	UID	Unsigned32	ID d'application du propriétaire
	4	Compressed	Boolean	Si, oui ou non, la compression s'applique à ce port
	5	TPDUsInOK	Unsigned32	Nombre de TPDU acceptées sur le port
	6	TPDUsInKO	Unsigned32	Nombre de TPDU rejetées sur le port
	7	TPDUsOutOK	Unsigned32	Nombre de TPDU envoyées sur le port
	8	TPDUsOutKO	Unsigned32	Nombre d'échecs d'émission de TPDU

11.6.2.5.4 Alertes

Le Tableau 235 décrit l'alerte pour indiquer une utilisation illégitime d'un port par une application.

Tableau 235 – Types d'alertes de l'objet de gestion de TL – Utilisation illégitime de port

Nom du type d'objet normalisé: TL management object (TLMO, objet de gestion de TL)					
Identificateur du type d'objet normalisé: 122					
Description de l'alerte: alerte pour indiquer une utilisation illégitime d'un port par une application					
Classe d'alertes (Enumerated: alarme ou événement)	**Catégorie d'alertes (Enumerated: diagnostic d'appareil, diagnostic de comm., sécurité ou processus)**	**Type d'alerte (Enumerated: en fonction de la catégorie d'alertes)**	**Priorité d'alertes (Enumerated: haut, moyen, faible, journal seulement)**	**Type de données de la valeur**	**Description de la valeur incluse avec l'alerte**
0 = Event	1= Communication diagnostic	0 =IllegalUseOfPort	8 = Medium	Type: Unsigned16	Numéro de port de 16 bits

Le Tableau 236 décrit l'alerte pour notifier une TPDU reçue qui adresse un port non enregistré.

Tableau 236 – Types d'alertes de l'objet de gestion de TL – TPDU reçue sur un port non enregistré

Nom du type d'objet normalisé: TL management object (TLMO, objet de gestion de TL)					
Identificateur du type d'objet normalisé: 122					
Description de l'alerte: alerte pour notifier une TPDU reçue sur un port non enregistré					
Classe d'alertes (Enumerated: alarme ou événement)	**Catégorie d'alertes (Enumerated: diagnostic d'appareil, diagnostic de comm., sécurité ou processus)**	**Type d'alerte (Enumerated: en fonction de la catégorie d'alertes)**	**Priorité d'alertes (Enumerated: haut, moyen, faible, journal seulement)**	**Type de données de la valeur**	**Description de la valeur incluse avec l'alerte**
0 = Event	1= Communication diagnostic	1=TPDUonUnregisteredPort	4 = Low	Type: OctetString	40 premiers octets de la TPDU

Le Tableau 237 décrit l'alerte pour notifier une TPDU reçue qui ne satisfait pas aux politiques de sécurité locales.

Tableau 237 – Types d'alertes de l'objet de gestion de TL – La TPDU ne concorde pas aux politiques de sécurité

Nom du type d'objet normalisé: TL management object (TLMO, objet de gestion de TL)					
Identificateur du type d'objet normalisé:					
Description de l'alerte: alerte pour notifier une TPDU qui ne satisfait pas aux politiques de sécurité locales					
Classe d'alertes (Enumerated: alarme ou événement)	**Catégorie d'alertes (Enumerated: diagnostic d'appareil, diagnostic de comm., sécurité ou processus)**	**Type d'alerte (Enumerated: en fonction de la catégorie d'alertes)**	**Priorité d'alertes (Enumerated: haut, moyen, faible, journal seulement)**	**Type de données de la valeur**	**Description de la valeur incluse avec l'alerte**
0 = Event	1= Communication diagnostic	2=TPDUoutOfSecurityPolicies	2 = Journal	Type: OctetString	40 premiers octets de la TPDU

12 Couche d'application

12.1 Généralités

La couche d'application (AL) définit des objets logiciels pour modéliser des objets du monde réel et définit également les services de communication nécessaires pour permettre la communication d'objet à objet entre les applications distribuées dans un environnement d'application ouvert et interopérable conforme à, et fonction de, la présente norme. La présente norme ne définit pas le fonctionnement des applications distribuées elles-mêmes; à savoir, ni le fonctionnement local de l'application elle-même, tel que la façon dont une application acquiert les valeurs des attributs d'objet qu'elle prend en charge pour l'accès ni les instructions concernant comment et quand une application applique les modèles et/ou les services définis dans le présent document ne sont adressés par la présente norme.

NOTE 1 Par exemple, une entrée analogique du monde réel est modélisée comme étant un objet AnalogInput. L'objet AnalogInput communique souvent sa variable de processus à une partie correspondante en utilisant le service d'édition fourni par l'AL.

L'AL prend en charge les appareils sans fil sur le terrain, ainsi que les passerelles qui intègrent un réseau sans fil conforme à la présente norme et ses appareils avec un système de commande d'hôte.[10]

Le modèle d'application dans la présente norme est spécifiquement conçu pour satisfaire aux contraintes d'environnements de communication sans fil.

NOTE 2 Une approche AL orientée objet est utilisée pour les raisons principales énoncées ci-après.

Les protocoles basés sur des commandes peuvent être conçus pour se conformer au modèle d'objet défini par la présente norme en décrivant les commandes comme étant des méthodes d'objet distinctes. A savoir, une application basée sur des commandes peut être modélisée en utilisant le modèle d'objet dans la présente norme.

Le modèle d'objet prend en charge des principes architecturaux bien admis de séparation logique des informations. Par exemple, les informations de gestion sont logiquement séparées des données de fonctionnement. Les informations opérationnelles pour des variables indépendantes sont logiquement séparées. Afin de maintenir la séparation d'informations, le protocole doit identifier l'objet correspondant. Cela ajoute un surdébit d'octets pour identifier l'objet, ce qui avait été jugé comme étant une approche plus que raisonnable pour la séparation architecturale d'informations.

12.2 Considérations relatives à l'énergie

Le besoin de prolonger la vie de batterie rend extrêmement importante la messagerie à haut rendement énergétique. L'utilisation de l'alimentation par batterie ou des techniques de récupération d'énergie pour les appareils de terrain connectés exige des considérations complémentaires dans la conception de la couche de communication, en comparaison à l'approche adoptée pour les appareils câblés. Non seulement chaque couche en communication a besoin de considérer la disponibilité des ressources d'appareil, mais également elle a besoin de considérer la réduction au maximum de la consommation d'énergie (dans des limites de contraintes d'architecture appropriées, naturellement) afin de prolonger la vie de la batterie ou de fonctionner dans les limites du budget de récupération.

Sachant que l'énergie est consommée par le traitement des messages, ainsi que par l'opération de commande fondamentale de l'appareil, l'efficacité des communications doit être équilibrée avec l'utilisation de principes architecturaux éprouvés et bien acceptés de séparation des informations ainsi qu'avec l'efficacité de traitement de message. Le modèle d'application native dans la présente norme est défini pour répondre à ces besoins.

12.3 Considérations de système de commande hérité

Des réseaux sans fil conformes à la présente norme peuvent être connectés à des systèmes de commande hérités.

NOTE 1 Un modèle qui s'intègre à des systèmes existants permet la réutilisation d'outils et d'interfaces existants et éprouvés et réduit également le temps global de développement et d'essai, ce qui produit des mises en œuvre robuste plus précoces.

Un processus d'application dans un appareil natif communique sur le réseau en utilisant seulement des services et des charges utiles définis par l'ASL. Un processus d'application dans un appareil non natif exige la communication de constructions qui ne sont pas définies par les charges utiles de service de l'ASL. Cette communication à partir de l'appareil non natif sur le réseau défini par la présente norme est accomplie en utilisant le sous-ensemble de

[10] Voir 5.2.6.5 pour un débat plus complet relatif aux rôles pris en charge par la présente norme.

services normalisés définis dans la présente norme qui prend en charge la communication de charges utiles qui ne sont pas explicitement et entièrement définies par la présente norme.

L'AL native définit des objets et services spéciaux pour soutenir la tunnellisation de protocoles non natifs pour répondre aux besoins d'intégration de systèmes.

Le jeu d'objets d'application définis qui prend en charge des charges utiles définies non natives est constitué de l'objet tunnel et l'objet d'interface. Le jeu de services d'application définis par une norme qui prennent en charge les charges utiles au-delà de celles définies dans la présente norme est constitué du service tunnel, qui est utilisé pour la communication apériodique, et du service édition utilisant le mode non natif, qui est utilisé pour des communications périodiques. Par exemple, la charge utile du service tunnel pour la messagerie apériodique afin d'encapsuler des données construites par un système hérité n'est pas définie dans le WISN.

NOTE 2 Une manière de réaliser une synthèse avec des systèmes existants est de créer une version à optimum d'énergie et à optimum de ressources d'une approche héritée existante orientée fils en établissant un mapping du modèle hérité au modèle de la présente norme et en utilisant directement l'AL native. De tels mappings doivent habituellement être définis par des consortiums de protocoles individuels, tels que la HART Communication Foundation (HCF), la Fieldbus Foundation (FF), PROFIBUS Nutzerorganisation (PNO), ODVA, qui prend en charge des protocoles gérés tels que CIP [Common Industrial Protocol (Protocole industriel commun)], et d'autres.[11] Une autre manière de réaliser la synthèse est d'utiliser un tunnel de protocoles à travers l'AL native. Pour les deux approches, les implications en termes d'énergie et de ressource sont des considérations importantes.

NOTE 3 Quelle que soit la méthode choisie, le système de plus haut niveau communique toujours avec les appareils sans fil au sein d'un réseau conforme à la présente norme.

NOTE 4 Des fichiers de description d'appareil électronique sont souvent utilisés pour satisfaire à cette exigence. Pour la présente norme, le langage de description d'appareil (DDL) ou le langage de description électronique de produit (EDDL) décrivent les appareils natifs.

NOTE 5 Voir Annexe R pour plus de détails concernant l'interface de système hôte.

12.4 Vue d'ensemble de la modélisation orientée objet

12.4.1 Généralités

Un modèle d'objet est un moyen neutre vis-à-vis de tout protocole, de toute plateforme et de tout langage de décrire et de distinguer les composants (éléments de système) qui ont une identité unique. Les objets séparent le monde en des morceaux sensés et gérables. Les définitions d'objet favorisent non seulement la modularité, mais aussi la réutilisabilité de composants. Un objet peut représenter toute chose qui a un état et un comportement; les objets exposent des attributs pour représenter l'état, et fournissent des méthodes qui opèrent sur l'état de l'objet pour effectuer des comportements particuliers. Par exemple, les méthodes spécifiques à un appareil qui peuvent être prises en charge par un DMAP peuvent inclure des méthodes d'autotest spécifiques à un appareil ou des méthodes de réinitialisation d'appareils spécifiques à un appareil.

12.4.2 Concept de communication d'objet à objet

Du point de vue de l'utilisateur, la communication d'AL se produit à partir d'un objet dans un processus d'application vers un autre objet dans un processus d'application. Les concepts de polymorphisme permettent aux mêmes techniques de communication d'être appliquées aux objets d'application utilisateur indépendants vis-à-vis de l'industrie selon la présente norme comme aux objets dépendants vis-à-vis d'une industrie selon la présente norme, ainsi qu'aux objets de gestion selon la présente norme.

---

[11] HART, FF-H1, PROFIBUS, et ODVA sont les marques commerciales de diverses organisations commerciales. Cette information est donnée à l'intention des utilisateurs de la présente norme et ne signifie nullement l'approbation ou la recommandation du propriétaire de la marque ou des produits associés. La conformité au présent document n'exige pas l'utilisation de l'appellation commerciale. L'utilisation des appellations commerciales requiert la permission du détenteur de l'appellation concernée.

En accord avec ce principe, le modèle d'application définit un modèle d'objet et un modèle d'interaction de communication (service et protocole). Le modèle d'application prend également en charge plusieurs processus d'application au sein d'un appareil, qui peuvent contenir chacun plusieurs objets normalisés. Cela permet à la présente norme de répondre aux besoins spécifiques du marché, tels que ceux pour les industries de transformation ou l'automation en usine, ainsi que de permettre la prise en charge pour les architectures d'appareils tant simple processeur que multiprocesseur.

L'objet d'application peut, par exemple utiliser les services de l'AL pour:

- lire la valeur d'un attribut d'un objet distant;
- écrire la valeur d'un attribut d'un objet distant;
- demander l'exécution d'une méthode spécifique à un objet d'un objet distant;
- rapporter une alerte relative à un objet distant;
- acquitter une alerte rapportée par un objet distant;
- éditer des données pour un objet distant en utilisant la largeur de bande de communication programmée;
- tunnelliser un message d'application natif non WISN vers un objet distant.

Le codage pour ces services peut être trouvé en 12.23.1.4.

12.4.3 Structure de la couche AL

L'AL est divisée en deux sous-couches, l'AL supérieure (UAL) et la sous-couche d'application (ASL), telle que montrée à la Figure 16. Il y a un rapport univoque (1:1) entre un ASLDE-SAP et un TLDE-SAP.

L'UAL contient les processus d'application pour l'appareil. Ces processus peuvent être représentés comme un UAP ou comme un processus de gestion (MP), comme le DMAP ou toute autre application de gestion logique telle qu'une application de gestion de sécurité.

Les UAP peuvent être utilisés, par exemple, pour:

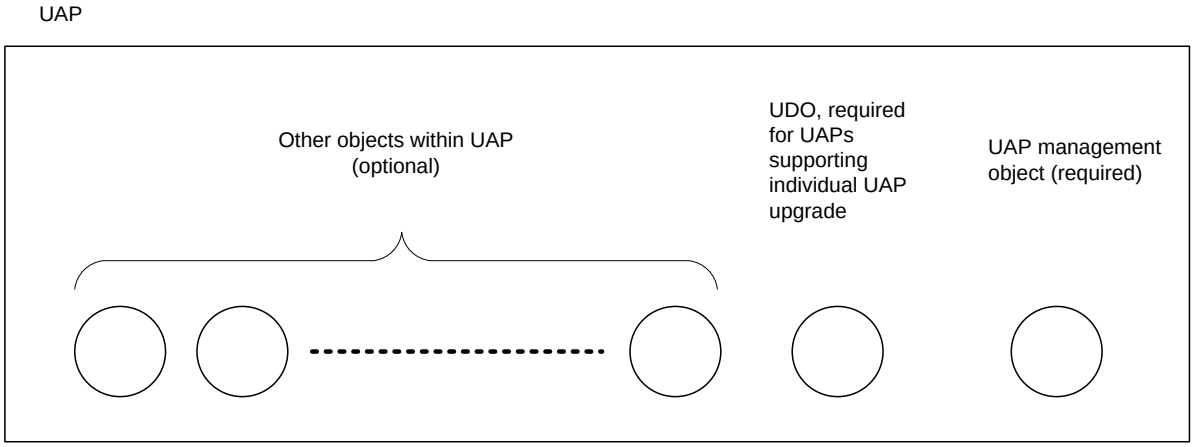
- traiter les matériels d'entrée et/ou de sortie;
- distribuer des communications vers un jeu d'UAP corésidents au sein d'un appareil (fonction proxy);
- prendre en charge la tunnellisation d'un protocole non natif (par exemple, hérité d'un système de commande) compatible avec l'environnement réseau de la présente norme; et/ou
- accomplir une fonction de calcul.

Un UAP peut accomplir une fonction individuelle ou n'importe quelle combinaison des fonctions. La manière dont un UAP accomplit ces fonctions en interne ne relève pas du domaine d'application de la présente norme. L'AL est concernée par le contenu de message spécifique à une application, le comportement visible de l'extérieur des objets normalisés contenus au sein de l'UAP, et les interfaces logiques à l'ASL qui représentent la communication d'UAP vers/depuis la suite inférieure de protocoles de communication de la présente norme. Les processus d'UAL peuvent contenir un ou plusieurs objets qui communiquent les uns avec les autres sur le réseau décrit dans la présente norme, en utilisant les services normalisés fournis par l'ASL.

NOTE La manière dont l'ASL met en œuvre son routage de messages interne ou sa communication entre applications au sein d'un appareil (au sein d'un UAP, au travers de plusieurs UAP dans un processeur commun, au travers de plusieurs UAP dans des processeurs physiques différents du même appareil, ou entre un UAP et un MP) relève d'une initiative locale et ne relève donc pas du domaine d'application de la présente norme.

12.4.4 Structure d'UAP

La Figure 110 décrit la structure globale d'un UAP tel que défini par la présente norme.



Légende

Anglais	Français
Other objects within UAP (optional)	Autres objets au sein de l'UAP (facultatif)
UDO, required for UAPs supporting individual UAP upgrade	UDO, requis pour les UAP prenant en charge la mise à niveau d'UAP individuel
UAP management object (required)	Objet de gestion d'UAP (requis)

Figure 110 – Objets d'application utilisateur dans un UAP

La représentation des applications et de leurs fonctions par des définitions normalisées d'objets permet la gestion et la construction uniformes d'applications distribuées. L'objet de gestion d'UAP identifie l'UAP auprès du réseau conforme à la norme et permet d'avoir une visibilité et/ou un contrôle sur certains aspects opérationnels de l'UAP en son ensemble. Cet objet de gestion d'UAP a un identificateur d'objet réservé de 1 (un). Si un UAP prend en charge la mise à niveau individuelle, l'UAP doit également contenir un objet UploadDownload (UDO) normalisé pour prendre en charge la mise à niveau d'UAP. Cet UDO a un identificateur d'objet réservé de 2 (deux). Si un UAP ne prend pas en charge la mise à niveau d'UAP individuelle, l'UAP ne doit pas contenir une instance d'objet avec l'identificateur d'objet de 2 (deux). Des objets complémentaires peuvent également être contenus au sein d'un UAP afin de fournir à l'UAP une fonctionnalité spécifique à une application.

NOTE 1 L'UAPMO est suffisant pour représenter l'UAP au système de communication.

NOTE 2 D'autres objets sont statiquement ou dynamiquement instanciés au sein de l'UAP.

NOTE 3 Il ne relève pas du domaine d'application de la présente norme de définir ce qui arrive aux données contenues au sein d'un UAP lorsque l'UAP est mis à niveau.

Le modèle d'interaction décrit la communication entre objets, y compris la classification de messages et les formats de messagerie. L'ASL contient les services qui prennent en charge la communication orientée objet et le routage vers l'objet de destination approprié au sein d'un UAP, à travers le réseau. Cette interaction réalise un mapping de l'ASL aux services fournis par la couche de suite inférieure de protocole de communication (voir Tableau 281 et Tableau 282 indiquant l'utilisation par l'AL des services de TL et les qualités de service). Entre l'UAL et l'ASL, il y a un SAP d'entité de données d'ASL (SAP d'ASLDE-n).

La gestion spécifique à l'ASL est également prise en charge localement par l'intermédiaire d'un SAP de gestion distinct.

12.5 Modèle d'objet

L'AL définie par la présente norme tire profit des concepts de modélisation orientés objet pour prendre en charge la tunnellisation tant de protocoles natifs que de protocoles (hérités) non natifs au sein d'applications. La tunnellisation de protocoles non natifs est réalisée par un UAP spécialisé qui inclut un ou plusieurs objets tunnel (TUN) et moyens de conversion de protocole. Cet UAP est constitué d'exactement un objet de gestion d'UAP pour permettre à la gestion uniforme de système et de réseau de l'UAP, plus un ou plusieurs TUN d'envoyer/recevoir des messages encapsulés tunnellisés vers l'UAP. D'autres objets natifs définis par la présente norme peuvent également être utilisés au sein d'un UAP de tunnellisation. Par exemple, la tunnellisation peut être utilisée pour la communication d'appareil sans fil à appareil sans fil, ainsi que la communication appareil sans fil à passerelle.

NOTE 1 Les constructions d'adressage autres que celles-ci ne relèvent pas du domaine d'application de la norme. Des constructions d'APDU de protocoles hérités (par opposition à des APDU natives) sont préservées au moyen d'encapsulation lorsque la tunnellisation d'application telle que définie par la présente norme est utilisée.

Une approche orientée objet est utilisée pour encapsuler les données (attributs) et la fonctionnalité (méthodes et état interne) pour la réutilisation et la cohérence. Les objets sont individuellement adressables en utilisant un identificateur d'objet qui est unique au sein de l'application. Cet identificateur unique permet à l'AL d'acheminer des messages vers la destination d'objet appropriée. Chaque message est interprété et traité par l'objet de destination en fonction du contexte et du contenu du message. Le fonctionnement d'objet est décrit en termes de fonctionnement, visible du réseau, de l'objet de destination qui est la cible du service d'AL.

La présente norme définit les identificateurs d'objet normalisés.

NOTE 2 Le complément des identificateurs normalisés pris en charge par un appareil est indiqué par la version de la présente norme qui est prise en charge. La version prise en charge est disponible auprès du DMAP. Voir Article 6 pour de plus amples détails.

Une instance d'objet qui est accessible au sein d'un processus d'application est distinguée d'une autre instance de la même classe d'objets dans le même processus d'application par son identificateur d'objet. Des instances d'objet différentes peuvent également avoir des valeurs d'attribut différentes et/ou des attributs conditionnels contenus dans son jeu d'attributs pris en charge.

A titre d'exemple, un processus d'application peut contenir deux instances de l'objet AnalogInput. Les instances d'objet au sein du processus d'application peuvent être distinguées par leur identificateur d'objet. Chaque objet peut prendre en charge différentes valeurs pour la mise à l'échelle et peut également exiger un différent complément d'alarmes devant être rapportées. Par conséquent, les instances peuvent avoir un complément différent d'attributs descriptor analogiques pris en charge par chaque instance.

Les objets définis dans les UAP et dans les MP de la présente norme adhèrent aux concepts traditionnels de modélisation d'objets; plus précisément, ces objets contiennent des attributs, et des méthodes appropriées spécifiques à un objet, si applicable, sont définies. L'AL définit des objets normalisés pour fournir l'interopérabilité (dans son domaine d'application) par l'intermédiaire de l'accès à des attributs normalisés et par l'intermédiaire de l'invocation de méthodes normalisées.

Des objets normalisés peuvent être classés dans des profils d'utilisation pour répondre aux besoins d'industries particulières.

Les objets de gestion normalisés sont censés être toujours disponibles. Des objets utilisateur d'application peuvent être instanciés de manière statique ou ils peuvent être instanciés de manière dynamique comme résultat d'une opération de téléchargement. Les objets normalisés sont extensibles selon les manières suivantes:

- Des types d'objets normalisés spécifiques à une industrie peuvent être ajoutés.
- Des types d'objet spécifiques à un fournisseur peuvent être ajoutés.
- Des attributs spécifiques à une industrie peuvent être ajoutés aux objets normalisés.
- Des attributs spécifiques à un fournisseur peuvent être ajoutés aux objets normalisés.
- Des méthodes spécifiques à une industrie peuvent être ajoutées à des objets par l'intermédiaire de profils spécifiques à une industrie.
- Des méthodes spécifiques à un fournisseur peuvent être ajoutées à des objets.
- Des profils spécifiques à une industrie de types d'objets peuvent être définis, tels qu'un profil d'industrie de transformation, un profil d'industrie d'automation en usine, et autres profils.

12.6 Modèle d'attribut d'objet

12.6.1 Généralités

Les objets définis par la présente norme prennent en charge deux sortes d'attributs, les attributs-clés d'objet et les attributs nommés.

Dans la présente norme, une ressource est représentée comme étant un objet et elle est identifiée par un attribut-clé d'objet, qui est une valeur numérique.

NOTE 1 La prise en charge d'un attribut-clé d'objet en utilisant une représentation alphanumérique de la ressource et celle des services d'annuaire correspondants pour localiser la ressource et résoudre la forme de nom alphanumérique externe en une forme numérique interne sont l'objet d'une normalisation future.

Un attribut spécifie un élément accessible représentant une propriété ou une caractéristique d'une ressource. Dans la présente norme, un attribut est représenté par une valeur numérique unique qui identifie de manière univoque l'attribut relativement à l'objet contenant. La plage prise en charge des valeurs valides pour un identificateur d'attribut est 0..4 095.

La présente norme définit les attributs d'objet normalisés. Des attributs normalisés complémentaires peuvent être définis à l'avenir.

NOTE 2 Le complément des attributs normalisés pris en charge est établi par la version de la présente norme qui est prise en charge. La version de la norme qui est prise en charge est disponible auprès du DMAP.

Le fait d'exposer les éléments de ressource comme étant des attributs d'un objet permet à l'état de l'objet d'être déterminé et permet également au comportement de l'objet d'être modifié. Un attribut de ressource est défini par:

- son influence sur le comportement d'objet;
- l'ensemble de valeurs qu'il peut prendre (telles que contraintes par la définition d'objet contenant l'attribut);
- les essais valides (par exemple, concordance d'ensemble de valeur valide) réalisables qu'il peut subir; et
- l'ensemble spécifique de conditions d'erreur qui peut découler d'un échec défini de l'objet en raison de l'accomplissement d'une opération orientée attribut.

Les attributs eux-mêmes n'ont pas de propriétés ou de sous-types accessibles. Des valeurs d'attribut peuvent être explicitement établies par des moyens externes ou par des moyens internes (par exemple, dérivées par calcul en utilisant les valeurs d'autres attributs).

Les attributs doivent avoir un type de données qui est un type scalaire normalisé défini par la présente norme. Les attributs peuvent avoir un type de données qui est une structure de données normalisée définie par la présente norme. Jusqu'à deux indices sont disponibles pour adresser les constructions avec des types normalisés définis par la présente norme. La plage valide pour un indice est 0..32 767.

NOTE 3 Par exemple, l'accès à un élément individuel d'une matrice unidimensionnelle de types scalaires normalisés est pris en charge. Comme autre exemple, l'accès à un élément individuel d'une structure de données contenue dans une matrice unidimensionnelle de telles structures est pris en charge.

Pour un attribut construit comme une matrice, la taille de la matrice doit être fixe et tous les éléments d'une matrice doivent être dimensionnés de façon homogène. Par exemple, une matrice de chaînes d'octets doit avoir la même taille de chaîne d'octets pour chaque élément de la matrice. Lorsque la dimension courante et la dimension maximale d'une matrice doivent être indiquées, il convient d'utiliser des informations de métadonnées. Ces informations de métadonnées sont décrites par le type de données de code 406 (structure de données Metadata_attribute) qui est défini en 6.2.6.3.

12.6.2 Attributs d'objets normalisés

Pour chaque objet normalisé, des attributs normalisés sont définis. Chaque attribut normalisé a un identificateur d'attribut normalisé qui est utilisé pour adresser l'attribut.

Les attributs d'objets normalisés peuvent être étendus selon les manières suivantes:

- Des attributs spécifiques à une industrie peuvent être ajoutés aux objets normalisés.
- Des attributs spécifiques à un fournisseur peuvent être ajoutés aux objets normalisés.

Des extensions aux attributs normalisés ont besoin d'être coordonnées pour s'assurer que les identificateurs d'attribut restent uniques au sein d'un type d'objet. Les mécanismes utilisés par des industries et des fournisseurs pour étendre les attributs de l'objet normalisé ne relèvent pas du domaine d'application de la présente norme.

12.6.3 Classification d'attributs

Les attributs sont classés pour fournir un guide relatif à leur fréquence de changement prévue. Ces informations sont utiles, par exemple, aux appareils passerelles qui placent des informations dans un cache. La fréquence à laquelle les valeurs d'attribut changent est caractérisée comme étant:

- "constant" (constante);
- "static" (statique);
- "static-volatile" (statique volatile);
- "dynamic" (dynamique); ou
- "non-cacheable" (non plaçable en cache).

Un attribut constant est invariable tout le temps. Un exemple d'un attribut constant est le numéro de série d'un appareil sans fil. Toutes les valeurs par défaut dans la présente norme sont des attributs constants. Les valeurs de ces attributs doivent être préservées lorsqu'un appareil subit un redémarrage à chaud/une panne de courant, lorsqu'un appareil se réinitialise aux valeurs d'usine par défaut ou lorsqu'une commande de réinitialisation donnée à l'attribut ou à l'objet de gestion pertinent est reçue.

L'information constante peut être:

- une information fixe qui ne change jamais, telle qu'une information pour indiquer un fabricant ou un numéro de série; et
- une information qui ne change pas pendant l'exploitation normale du système, mais qui peut changer, par exemple, lorsqu'un téléchargement de firmware se produit.

Un attribut "static" change rarement sa valeur. Habituellement, les données statiques sont changées comme résultat d'un message, d'une demande, ou d'un événement externe. Certaines informations statiques peuvent, par exemple, être seulement changées par un outil de configuration. Les plages de fonctionnement, les unités, les points d'extrémité de communication, les alarmes, et les valeurs d'entrée constantes sont des exemples

d'informations statiques. Les attributs stockant de l'information relative à la mise en service, ainsi que les informations de configuration fournies à un appareil, sont des attributs statiques. Les valeurs de ces attributs doivent être préservées lorsqu'un appareil subit un redémarrage à chaud/une panne de courant ou lorsqu'une commande de réinitialisation à un attribut ou objet de gestion sans rapport est reçue.

Un attribut " static-volatile" change rarement sa valeur. Habituellement, les données statiques volatiles sont changées comme résultat d'un message, d'une demande, ou d'un événement externe. Certaines informations statiques volatiles peuvent, par exemple, être seulement changées par un outil de configuration. Les valeurs de ces attributs peuvent ne pas être préservées lorsqu'un appareil subit un redémarrage à chaud/une panne de courant. Les valeurs de ces attributs doivent être préservées lorsqu'une commande de réinitialisation à un attribut ou objet de gestion sans rapport est reçue.

Un attribut "dynamic" peut être changé spontanément par l'appareil contenant l'objet et sans stimulation externe émanant du réseau sans fil. Des exemples d'attributs dynamiques sont les valeurs changeant fréquemment, telles que les variables de processus, les calculs et les temporisateurs. Des attributs dynamiques peuvent être traités différemment par un cache de passerelle changeant rarement des valeurs (statiques); cela appartient entièrement à la mise en œuvre interne de la passerelle. Un attribut dynamique n'est pas tenu de survivre lorsqu'un appareil subit un redémarrage à chaud/une panne de courant ou lorsqu'un appareil se réinitialise aux valeurs d'usine par défaut. Il peut être réinitialisé lorsqu'une commande de réinitialisation à un attribut ou objet de gestion pertinent est reçue.

Les informations "non-cacheable" ne sont jamais mises en tampon; par exemple, elles peuvent être utilisées pour l'information critique telle que l'information relative à la sécurité (dont l'utilisation peut ne pas relever du domaine d'application explicite de la présente norme), et pour les valeurs qui changent trop souvent pour faire du placement en cache une technique valide. Chaque fois que la valeur d'un attribut "non-cacheable" est demandée, elle doit être récupérée dans l'appareil final qui possède l'objet et l'attribut.

Si le placement en cache local d'appareil est nécessaire, cela relève de la responsabilité locale de l'application.

12.6.4 Accessibilité d'attribut

Les attributs accessibles du réseau peuvent être:

- accessibles pour être lus seulement; ou
- accessibles pour être tant lus qu'écrits.

12.7 Modèle de méthode

Les méthodes représentent le jeu d'interfaces (fonctions) spécifiques à un type d'objet qui peuvent être utilisées pour accéder à une instance d'objet. Par exemple, l'objet UploadDownload prend en charge une méthode StartUpload.

Les méthodes d'objets normalisés doivent toujours être disponibles, et ne doivent pas être renforcées au-delà de la définition donnée par la présente norme.

Les méthodes ne doivent pas être définies si le résultat équivalent peut être obtenu en utilisant un service d'AL (indépendant vis-à-vis de tout type d'objet) normalisé (le service lecture fourni par l'ASL, par exemple). La définition d'une méthode peut être justifiée, par exemple, pour remplacer une séquence de transactions de communication afin d'économiser de l'énergie. La définition d'une méthode peut également être justifiée lorsque des questions de synchronisation peuvent résulter si des actions individuelles sont utilisées plutôt qu'un ensemble de transactions atomiques.

Des méthodes d'objets normalisés peuvent être ajoutées à l'avenir par la présente norme.

Le déclenchement à base de temps des activités de processus d'application n'est pas une responsabilité de sous-système de communication. Si un tel déclenchement à base de temps est nécessaire, il est permis d'utiliser soit un paramètre d'une méthode, soit un attribut dédié d'un objet. Si la coordination à travers des objets est exigée, un objet d'application peut être défini avec un attribut représentant l'action coordonnée, agissant comme un proxy pour la coordination.

12.8 Modèle d'alerte

Le terme alerte est utilisé pour décrire un message d'application qui conseille ou avertit le destinataire de la présence d'une situation d'intérêt imminente ou existante. Le modèle d'alerte décrit des alertes rapportées par les objets résidents de processus d'application et le mécanisme pour les rapporter.

Deux types d'alertes sont pris en charge, à savoir les alarmes ("alarm") et les événements ("event"). L'événement est le terme utilisé pour représenter une condition sans état (c'est-à-dire, pour indiquer qu'une situation s'est produite). Les événements rapportent simplement que quelque chose s'est produit. Une alarme est un état avec état d'une situation existante, par exemple, qu'une alarme est passée à un état anormal, ou est revenue à la normale à partir d'un état anormal. La condition d'alarme reste true jusqu'à ce que la condition d'alarme soit éliminée. Alert est le terme utilisé pour décrire la messagerie relative à une condition d'événement ou une condition d'alarme. Les alarmes et les événements sont rapportés par l'intermédiaire du mécanisme de rapports d'alerte défini dans la présente norme.

Une alarme est caractérisée par un état, et les alertes sont utilisées pour rapporter:

- l'occurrence d'une condition; et
- le retour à la normale de la condition précédemment rapportée.

Les événements et les alarmes pris en charge par la présente norme s'inscrivent dans l'une des catégories suivantes:

– relative à un appareil;
– relative à la communication;
– relative à la sécurité; ou
– relative au processus de commande.

Chaque alarme et événement définis pour un objet doivent avoir un attribut associé qui décrit comment il(elle) est rapporté(e). Cet attribut associé doit inclure:

- s'il est activé ou désactivé pour les rapports; et
- sa priorité (importance).

Pour toutes les alarmes, l'information descriptive doit également inclure, dans le cas où l'alerte est une alarme, si, oui ou non, la condition est en alarme ou hors alarme.

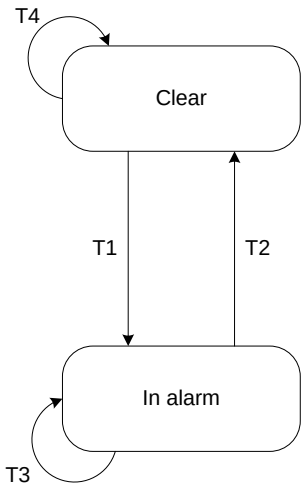
Une alarme analogique se produit lorsqu'une valeur atteint ou dépasse une limite établie. Pour les alarmes de valeur analogique, l'information descriptive doit inclure également l'information relative à la limite, le cas échéant, concernant le moment où la condition d'alarme est déclenchée.

12.9 Modèle d'état d'alarme

Le Tableau 238 et la Figure 111 représentent le modèle d'état d'alarme.

Tableau 238 – Table d'états pour des transitions d'alarme

Transition	Etat actuel	Evénement(s)	Action(s)	Etat suivant
T1	Clear	Alarme détectée	Rapporter l'alarme à l'ARMO dans le DMAP	In alarm
T2	In alarm	Clear est détecté	Rapporter alarm clear à l'ARMO dans le DMAP	Clear
T3	In alarm	Recovery requested	Rapporter l'alarme à l'ARMO dans le DMAP	In alarm
T4	Clear	Recovery requested	Ignorer	Clear
NOTE La récupération est habituellement demandée par un appareil distant.				



Légende

Anglais	Français
Clear	Sans
In alarm	En alarme

Figure 111 – Modèle d'états pour alarme

La détection d'alarme s'applique aux valeurs tant analogiques que discrètes. Les exemples d'alarmes analogiques incluent:

- alarmes de limite analogiques (par exemple, lorsqu'une valeur excède un seuil haut ou bas);
- alarmes d'écart analogiques (par exemple, la différence entre une variable de processus et un point de consigne);
- une alarme Boolean (par exemple, lorsque l'état du Boolean concorde avec le paramètre de limite discrète); et
- les diagnostics, tels que ceux définis dans la recommandation NAMUR NE107.

NOTE 1 Les alarmes qui dépendent de l'évaluation de conditions d'état spécifique à un appareil, spécifique entre objets ou spécifique au sein d'un objet sont considérées comme relevant d'une initiative locale et ne relèvent donc pas du domaine d'application de la présente norme.

NOTE 2 Des niveaux différents de conditions d'alarme sont indiqués par des alarmes différentes. Par exemple, pour une entrée analogique, une alarme High représente un niveau, et une alarme High-High représente un niveau supérieur.

12.10 Modèle d'état d'événement

12.10.1 Généralités

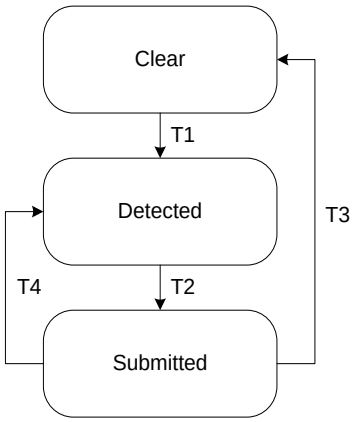
Le modèle d'état pour un événement est un sous-ensemble du modèle d'état pour une alarme.

12.10.2 Table et transitions d'états

Le Tableau 239 et la Figure 112 représentent respectivement la table d'état de l'événement et les transitions d'états de l'événement.

Tableau 239 – Table d'états pour des transitions d'événement

Transition	Etat actuel	Evénement(s)	Action(s)	Etat suivant
T1	Clear	Une condition d'événement est détectée	Déterminer les caractéristiques des rapports (la priorité, par exemple)	Detected
T2	Detected	Une condition d'événement est rapportée	Rapporter l'événement à l'ARMO dans le DMAP	Submitted
T3	Submitted	Présentation de la condition d'événement à l'ARMO comme achevée	Réinitialiser pour se préparer pour le prochain rapport d'événement	Clear



Légende

Anglais	Français
Clear	Sans
Detected	Détecté
Submitted	Présenté

Figure 112 – Modèle d'événement

12.11 Rapports relatifs à l'alerte

12.11.1 Généralités

Des alertes sont rapportées promptement et exactement horodatées en utilisant une communication de rapports d'alerte unidirectionnelle placée en file d'attente. Les rapports relatifs aux alertes unidirectionnelles placées en file d'attente impliquent que l'appareil détectant l'alerte rapporte la condition en utilisant un flux de communication source/puits. Un acquittement d'alerte unidirectionnelle placée en file d'attente est reçu en retour.

NOTE 1 Dans un message édité, l'information de statut indique parfois qu'une alerte est disponible dans l'appareil rapporteur, accessible par l'intermédiaire du service lecture C/S. Cette méthode est parfois utilisée pour

l'automation en usine; d'autres systèmes d'automation en usine éditent une étiquette à un serveur qui génère des alarmes en soumettant à essai les valeurs limites dans le serveur.

NOTE 2 La communication source/puits est utilisée plutôt que client/serveur par anticipation d'une version future de la présente norme qui prend en charge des rapports alertes de multidiffusion.

12.11.2 Types d'alerte

L'objet de gestion de rapports d'alerte (ARMO) contenu dans le DMAP fournit l'encapsulation du rapport d'alerte, traite des temporisations et des répétitions de tentative, et étrangle les rapports d'alerte d'une façon commune pour toutes applications contenues au sein de l'appareil.

Il ne doit y avoir qu'un seul ARMO dans chaque appareil, dans le DMAP. Comme décrit en 6.2.7.2.3, le DMAP peut fournir un accès limité à des entités autres que le SMAP afin de prendre en charge des services relatifs aux alertes relatives à un processus et les alertes relatives à un appareil. Des acquittements d'alertes doivent être adressés à l'ARMO.

Les alertes de diagnostics sont spécifiques à l'appareil les rapportant. Par exemple, les alertes de diagnostics peuvent indiquer

- qu'une erreur s'est produite;
- qu'un symptôme a été détecté, qui peut indiquer qu'une erreur non diagnostiquée s'est produite;
- qu'un symptôme a été détecté, qui peut indiquer qu'une erreur peut se produire dans le futur;
- qu'une erreur se produira si une action préventive n'est pas entreprise.

Les alertes de diagnostics peuvent concerner un appareil dans son ensemble, un composant individuel, ou un jeu défini de composants d'un appareil. Les alertes de diagnostics peuvent être sans état ou orientées état. Les alertes de diagnostics peuvent être spécifiées par la présente norme, comme pour les alarmes relatives à une communication, ou peuvent être spécifiques à un fournisseur. Les alertes de diagnostics fournissent les informations qui peuvent être ultérieurement examinées pour établir des profils de comportement de systèmes de communication et/ou d'appareils.

Les alertes de processus sont spécifiques au processus commandé par l'appareil rapportant l'alarme de processus. Une alarme de processus indique une situation dans laquelle la variable signalée par une alarme a dépassé les limites opérationnelles établies. Par exemple, une alerte de processus peut être générée lorsqu'une condition de commande mesurable se situe hors des paramètres désirés de fonctionnement du système de commande.

Les alertes de processus fournissent des informations qui peuvent être ultérieurement examinées pour établir des profils de comportement de systèmes de commande, telles que:

– alertes qui se produisent souvent dans une séquence particulière;
– alertes qui se produisent souvent en étant proches dans le temps;
– alertes qui furent actives pendant des périodes de temps significatives;
– actions qui doivent résoudre une situation d'alerte;
– assistance pour déterminer les valeurs de réglages optimales du point de déclenchement et de l'hystérésis; ou
– l'information concernant les performances des systèmes de commande en termes de prévention et de résolution des alertes.

Les alertes de processus concernent un objet de commande particulier et une valeur d'attribut de l'objet en question (par exemple, le PV d'un objet d'entrée analogique). Les alertes de processus sont habituellement orientées état (c'est-à-dire des alarmes).

Un octet est utilisé pour coder les informations du type d'alerte. Pour tous les objets, trois plages normalisées sont identifiées pour les définitions disjointes du type d'alerte spécifique.

00..49: réservé pour et défini par la présente norme;

51..100: réservé pour des profils d'industrie d'une norme future;

101..255: attribuable au fournisseur pour les alertes spécifiques aux fournisseurs

12.11.3 Informations relatives aux rapports d'alerte

L'en-tête d'APDU indique l'application et l'objet qui ont initié la communication. Pour des rapports d'alerte, cela indiquerait le DMAP et l'ARMO. L'information de rapports d'alerte individuelle doit donc également identifier le processus d'application et l'objet au sein de ce processus qui est la source détectrice de l'alerte. En plus, les rapports d'alerte doivent inclure les informations suivantes:

- temps réseau de la détection;
- identificateur d'alerte individuel (de sorte que des doublons peuvent être détectés par le processus d'UAL);
- classe d'alertes (alarme ou événement);
- sens d'alarme (transition entrant en alarme, ou pas (c'est-à-dire soit retour à la normale soit un événement));
- catégorie d'alerte (diagnostic d'appareil, relative à une communication, relative à la sécurité, ou relative à un processus);
- priorité alerte (plages définies pour les priorités "high", "medium", "low", et "journal-only");
- type d'alerte (spécifique à un objet, et au sein de la catégorie d'alertes spécifique);

NOTE Voir 6.2.7.2 pour plus d'informations sur les alertes de communication et 6.1.2 pour plus d'informations sur les alertes de sécurité.

- taille des données associées en octets; et
- données associées pour la condition d'alerte.

Il convient que les données associées pour les diagnostics d'appareil soient définies pour la compatibilité avec la recommandation NAMUR NE107; il convient que de tels diagnostics indiquent si la condition de l'appareil est anormale, et le cas échéant la classe NAMUR: échec, ne relevant pas de la spécification, maintenance de diagnostic ou fonction de commande de diagnostic.

12.11.4 Récupération sur état d'alarme

En cas de perte de communication avec un appareil sans fil, les industries de transformation exigent que les conditions existantes soient récupérables par un appareil de réception d'alerte, comme en déterminant l'état lorsqu'un appareil de réception d'alerte démarre.

NOTE 1 Plusieurs conditions d'alarme peuvent coexister dans un appareil de commande de processus.

La récupération d'état d'alarme peut être demandée de l'ARMO. Une seule demande de récupération d'alarme déclenche une nouvelle production de rapports relatifs à toutes les conditions d'alarme existantes dans l'appareil d'une catégorie donnée. Lors de la récupération d'alarmes, l'appareil rapportant les alarmes doit fournir des alertes pour indiquer le moment où la récupération débute et le moment où la récupération est achevée.

NOTE 2 Les conditions d'événements étant sans état, elles ne sont donc pas récupérables.

12.12 Modèle d'interaction de communication

12.12.1 Généralités

La communication native dans la présente norme prend en charge à la fois le protocole natif et l'encapsulation de protocoles hérités par l'intermédiaire de la tunnellisation. Les types suivants de flux de communication sont anticipés pour les appareils conformes:

- communication unidirectionnelle en file d'attente (exemple: production de rapports d'alarme ou acquittement d'alarme);
- communication bidirectionnelle en file d'attente (exemple: lecture, écriture, invocation de méthode); et
- communication de publication unidirectionnelle en file d'attente (exemple: édition). L'emplacement réel des tampons utilisés pour contenir les données pour la communication de publication unidirectionnelle programmée est une question locale d'appareil.

NOTE Les publications de données programmées placées en file d'attente (publication périodique, publication de changement d'état, et publication commandée par une application) se produisent toutes (selon les besoins) dans la phase programmée. Les contrats de communication pour la communication périodique utilisent la communication de publication unidirectionnelle placée en file d'attente. Les contrats de communication pour la communication apériodique utilisent un paradigme de communication placée en file d'attente.

12.12.2 Communication de publication unidirectionnelle placée en file d'attente

12.12.2.1 Généralités

La communication unidirectionnelle en file d'attente est utilisée quand une application éditrice envoie un message vers une application abonnée. Le tampon contient le message à envoyer. Sur chaque contrat de publication unidirectionnelle en file d'attente, il y a un paramètre pour indiquer si le tampon doit être toujours émis (que le contenu ait été mis à jour ou pas), ou si le tampon doit être émis seulement s'il a été mis à jour depuis l'émission antérieure.

12.12.2.2 Contenu de tampon toujours émis

Dans la communication unidirectionnelle placée en file d'attente, si une suite de protocoles de communication d'édition reçoit une autre demande de service Publish ASL pour un contrat de communication particulier avant que le message précédent n'ait été émis, la nouvelle demande remplace la demande précédente. Dans l'abonné, si un nouveau message est reçu avant que le précédent n'ait été livré à l'application, le nouveau message doit remplacer le message non délivré précédent.

NOTE 1 Pour établir un contrat pour la communication périodique, le gestionnaire de système veille à ce que la capacité des appareils intermédiaires le long d'un chemin est adéquate pour prendre en charge la communication périodique.

NOTE 2 Il est anticipé qu'une application qui reçoit une publication plus ancienne après une plus récente peut choisir de rejeter la publication plus ancienne.

Si une suite de protocoles de communication d'édition ne reçoit pas une nouvelle demande de service pour le contrat lorsqu'il est temps d'émettre, la demande précédente doit être réémise. Si l'abonné reçoit la même demande de service d'application en succession, l'application abonnée doit traiter cette situation comme une réception d'un message en doublon. Le traitement d'application d'un message en doublon placé en file d'attente est laissé à l'application, et n'est pas défini par la présente norme.

12.12.2.3 Contenu de tampon émis en cas de changement seulement

Ce mode de communication placée en file d'attente prend en charge un mécanisme de communication de changement d'état.

Dans la communication unidirectionnelle en file d'attente, si une suite de protocoles de communication d'édition reçoit une demande de service pour un contrat de communication

avant que le message précédent n'ait été émis, la nouvelle demande remplace la demande précédente. Dans l'abonné, si un nouveau message est reçu avant que le précédent n'ait été livré à l'application, le nouveau message doit remplacer le message non délivré.

NOTE Pour établir un contrat pour la communication périodique, le gestionnaire de système veille à ce que la capacité des appareils intermédiaires le long d'un chemin est adéquate pour prendre en charge la communication périodique.

Si une suite de protocoles de communication d'édition ne reçoit pas une nouvelle demande de service pour le contrat lorsqu'il est temps d'émettre, la demande précédente ne doit pas être réémise. Si l'abonné reçoit la même demande de service d'application en succession, l'application abonnée doit traiter cette situation comme une situation d'erreur. Le traitement d'application d'un message en doublon placé en file d'attente est laissé à l'application, et n'est pas défini par la présente norme.

12.12.3 Communication unidirectionnelle placée en file d'attente

La communication unidirectionnelle en file d'attente prend en charge la distribution placée en file d'attente de services de communication ASL non confirmés (unidirectionnels). Pour répondre à ce type de besoin de communication, les couches inférieures des correspondants sont censées fournir un service de transfert de données placé en file d'attente.

Le traitement d'application d'un doublon d'un AlertReport doit entraîner l'envoi d'un autre AlertAcknowledgement. La réception d'un doublon d'un AlertAcknowledgement doit être ignorée. Le traitement d'application de doublons de messages Tunnel unidirectionnels placés en file d'attente est laissé à l'application, et n'est pas défini par la présente norme.

12.12.4 Communication bidirectionnelle placée en file d'attente

12.12.4.1 Généralités

La communication bidirectionnelle placée en file d'attente prend en charge la distribution placée en file d'attente de services de communication ASL confirmés (bidirectionnels). Pour répondre à ce type de besoin de communication, les couches inférieures des correspondants sont censées fournir un service de transfert de données placé en file d'attente.

Ce nombre maximal de demandes de service confirmées bidirectionnelles en file d'attente (client/serveur) simultanément en cours autorisées pour un contrat est indiqué au processus d'application par le contrat de communication lorsque le contrat est accordé. La valeur par défaut pour cette valeur maximale doit être 1, c'est-à-dire que la valeur par défaut indique que le contrat prend en charge une seule demande en cours à un moment quelconque.

Le traitement d'application d'un doublon de demande est d'envoyer une autre réponse. Le traitement d'application d'un doublon de réponse lorsqu'il est reçu une réponse qui ne concorde pas avec un identificateur de demande en cours doit être d'ignorer la réponse.

12.12.4.2 Répétitions de tentative et contrôle de flux

12.12.4.2.1 Généralités

L'AL définie par la présente norme doit suivre à la trace ce qui arrive aux demandes de service client/serveur placées en file d'attente qu'elle envoie. Cela est nécessaire pour deux raisons, pour assurer la fiabilité de la livraison et pour le contrôle de flux.

Pour la fiabilité de la livraison, l'application a besoin de pouvoir déterminer quand il convient qu'elle renvoie (répète) le message. Il y a deux situations dans lesquelles la répétition peut être nécessaire: la première quand la demande de message n'est pas arrivée à destination et la seconde quand la demande de message est arrivée, mais que la réponse de l'application n'a pas réussi à revenir au demandeur original. Le contrôle de flux est nécessaire pour vérifier que l'appareil de destination n'est pas submergé de messages qu'il ne peut pas traiter, et aussi pour protéger le réseau et optimiser le débit de réseau.

La présente norme prend en charge la notification d'encombrement explicite directe par la suite de protocoles de communication inférieure et prend en charge un écho de niveau d'application de retour à un client de demandeur de service à quatre parties si l'encombrement s'est produit sur le chemin pris pour la demande initiale de service.

Pour permettre simultanément plusieurs demandes en cours tout en permettant toujours à une application de réaliser à la fois la livraison fiable et le contrôle de flux, chaque demande de client doit contenir un identificateur unique. La réémission d'une demande (répétitions de tentative) doit utiliser le même identificateur. Cet identificateur permet à l'application de mettre en œuvre une technique de fenêtre glissante pour contrôler le flux. Le client doit démarrer un temporisateur de temporisation relatif au service lorsqu'il initie une demande de service de client. Ce temporisateur doit être basé sur les temps de propagation aller-retour (RTT) pour des messages, et il doit accorder suffisamment de temps pour qu'un message issu d'une application dans l'appareil X atteigne l'application de destination dans l'appareil Y, pour que l'application de serveur dans l'appareil Y produise une réponse, et pour que la réponse revienne à l'application demandeuse de service dans l'appareil X.

NOTE Cette méthode est communément connue dans les communications comme étant la communication utilisant un acquittement positif avec réémission.

La Figure 113 présente un exemple de trois messages de demande d'écriture en cours simultanées avec un seul message concaténé qui contient les réponses de toutes les demandes d'écriture en cours. La concaténation est utilisée pour économiser du trafic de messagerie.

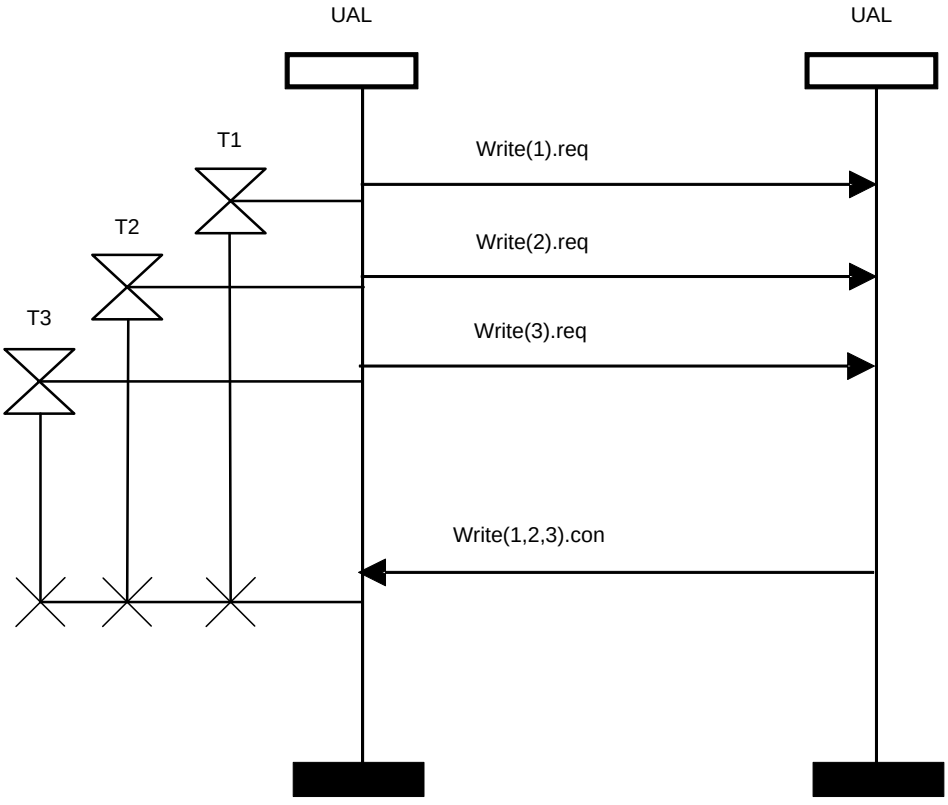


Figure 113 – Exemple réussi de plusieurs demandes en cours avec concaténation des réponses

12.12.4.2.2 Répétitions de tentative et intervalles de temporisation

12.12.4.2.2.1 Généralités

La méthode définie par l'IETF RFC 6298 doit être utilisée pour calculer une valeur appropriée pour l'intervalle de temporisation de répétitions de tentative (RTO). Pour calculer le RTO

courant, un client doit maintenir deux variables d'état, $SRTT$ temps de propagation aller-retour lissé) et $RTTV$ (variation du temps de propagation aller-retour), dans la portée d'un contrat.

Jusqu'à ce qu'une mesure du temps de propagation aller-retour (RTT) ait été faite pour un segment envoyé entre le client et le serveur, il convient que le client mette RTO = 3 s.

Lorsque la première mesure R de RTT est faite, le client doit mettre:

$SRTT = R$

$RTTV = R/2$

$RTO = SRTT + 4 \times RTTV$

Quand un RTT ultérieur est disponible, R est effectuée. Le client doit mettre à jour le $RTTV$, le $SRTT$, et le RTO en utilisant les calculs suivants, où la valeur recommandée pour β est 0,25, et la valeur recommandée pour α est 0,125:

$RTTV = (1 - \beta) \times RTTV + \beta \times |SRTT - R|$

$SRTT = (1 - \alpha) \times SRTT + \alpha \times R$

$RTO = SRTT + 4 \times RTTV$

Chaque fois que RTO est calculé, le RTO doit être arrondi selon les règles suivantes:

Si RTO < 1 s, mettre RTO = 1 s.

Si RTO < 60 s, mettre RTO = 60 s.

La détermination de l'occurrence de temporisation relève d'une initiative locale. Lorsqu'une temporisation a été déterminée comme s'étant produite, le repli exponentiel doit être utilisé pour les temporisations consécutives en mettant $RTO = RTO \times 2$ pour envoyer les répétitions de tentative. Il convient que la valeur maximale de 60 s soit utilisée pour fournir une limite supérieure à cette opération de doublement. Les répétitions de tentative cessent soit lorsqu'une réponse est reçue, soit lorsque la limite maximale de répétitions de tentative est atteinte. Le nombre maximal de répétitions de tentative autorisé pour une demande de client est indiqué par l'intermédiaire d'un attribut de l'objet de gestion UAP. La valeur sélectionnée pour le nombre maximal de répétitions de tentative autorisé relève d'une initiative locale.

NOTE L'IETF RFC 6298 contient une recommandation concernant la gestion du temporisateur TimeoutInterval.

12.12.4.2.2.2 Répétitions de tentative pour des messages sans ordre

Les messages sans ordre sont indépendants en ce que les réponses peuvent être reçues dans un ordre différent de l'ordre dans lequel les demandes ont été envoyées. En conséquence, chaque message de demande temporise et fait l'objet d'une répétition de tentative de manière indépendante.

La Figure 114 est un exemple de la façon dont une temporisation et une répétition de tentative du deuxième message dans une séquence de trois messages sans ordre, en raison de l'échec de la demande à atteindre le serveur, sont traitées.

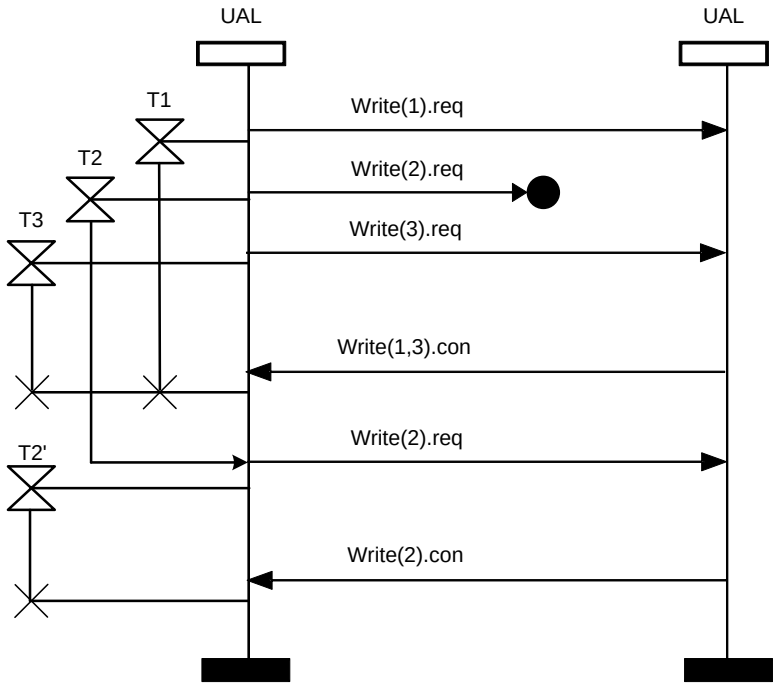


Figure 114 – Exemple de plusieurs demandes sans ordre en cours avec deuxième demande d'écriture initialement infructueuse

12.12.4.2.2.3 Répétitions de tentative pour des messages ordonnés

Les messages ordonnés sont dépendants de l'ordre; à savoir, les réponses ne peuvent pas être reçues dans un ordre différent de l'ordre dans lequel les demandes ont été envoyées. En conséquence, si un message plus tardif reçoit une réponse avant un message plus précoce, cela indique que le message pour lequel aucune réponse n'avait été reçue doit être temporisé et faire l'objet d'une répétition de tentative.

La Figure 115 est un exemple de la façon dont une temporisation et une répétition de tentative du deuxième message dans une séquence de trois messages sans ordre, en raison de l'échec de la demande à atteindre le serveur, sont traitées.

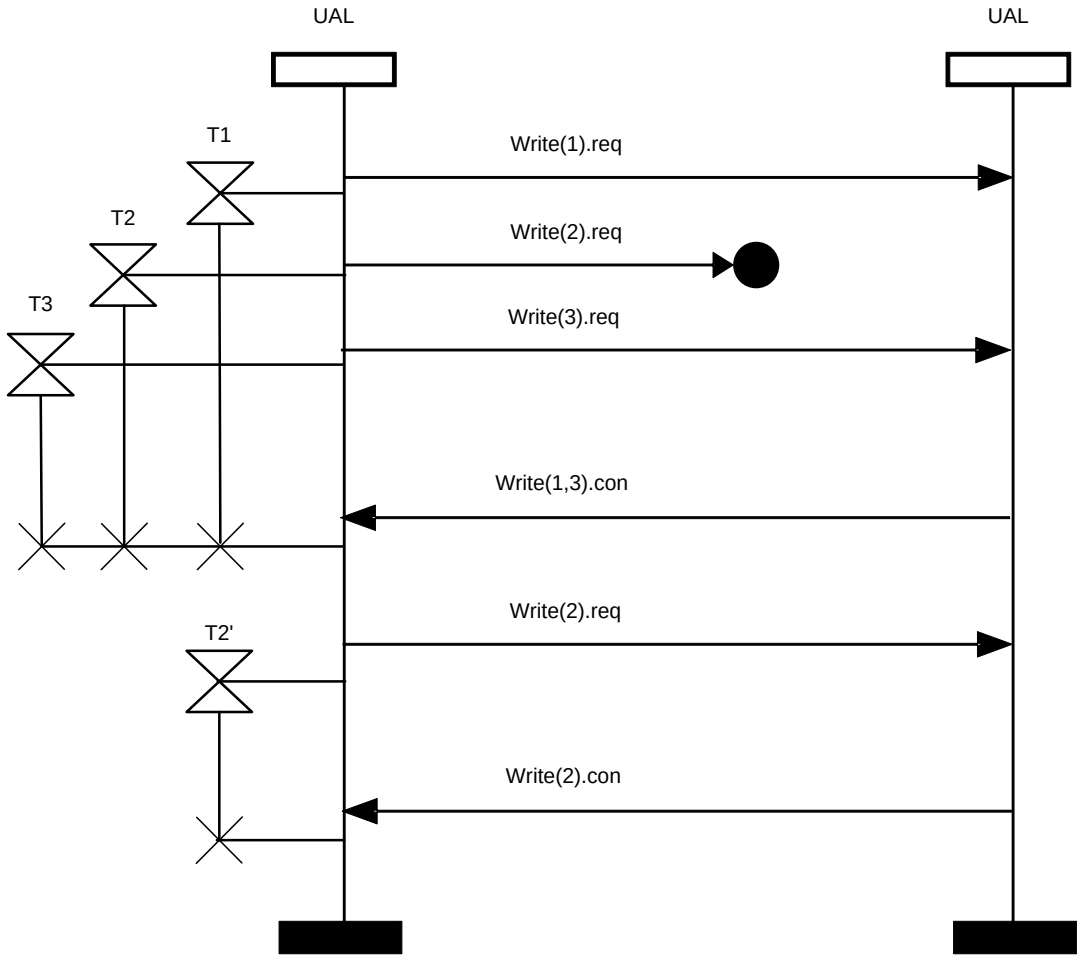


Figure 115 – Exemple de plusieurs demandes ordonnées en cours avec deuxième demande d'écriture initialement infructueuse

La livraison ordonnée concerne seulement le téléchargement montant/téléchargement descendant. Le Max_Send_Window_Size pour des contrats de communication de téléchargement montant/téléchargement descendant doit être fixé à 1. A ce titre, la livraison de messages ordonnée n'est pas prise en charge par les couches inférieures de la suite de protocoles définie par la présente norme.

12.12.4.2.3 Contrôle de flux

Les communications de client/serveur ne sont pas contrôlées par le débit de processus d'application; l'équité de débit d'AL est plutôt mise en application par les processus d'application sur une base par contrat afin de réduire au maximum l'équité de coût de l'encombrement sur le réseau.

Max_Send_Window_Size (Tableau 26) pour un contrat est le nombre maximal de demandes client qui peuvent simultanément être en attente d'une réponse dans la portée d'un contrat. Il convient (mais cela n'est pas exigé) que les clients utilisent des identificateurs de demande séquentiellement contigus. La valeur de Max_Send_Window_Size est établie sur une base contractuelle par le gestionnaire de système. L'OutstandingList représente les messages qui ont été envoyés et qui attendent actuellement une réponse.

L'AvailableSendWindowSize représente la fenêtre d'envoi utilisable, à savoir, l'ensemble de demandes client qui peuvent être envoyées, sans violer le Max_Send_Window_Size, en prenant en compte le nombre de messages contenus dans l'OutstandingList. Lorsque les fenêtres sont vides, et CurrentSendWindowSize est égal à Max_Send_Window_Size, la fenêtre d'envoi utilisable s'étire à partir de la dernière demande client acquittée pour le

prochain nombre Max_Send_Window_Size de demandes, et représente le jeu des demandes client qui peuvent être envoyées, sans violer le Max_Send_Window_Size.

Les applications doivent initialement mettre leur valeur pour que leur CurrentSendWindowSize soit un (1). Le RTT est alors mesuré par l'application pour n transactions complètes. S'il ne se produit aucune temporisation au cours de ces transactions, et si le Max_Send_Window_Size n'a pas été atteint, le CurrentSendWindowSize doit être incrémenté de un (1). La valeur de CurrentSendWindowSize doit être égale à la taille de la CurrentSendWindowLimit +1.

NOTE Ce mécanisme pour augmenter la taille de fenêtre est plus conservateur que celui habituellement utilisé pour répondre à des exigences d'évitement d'encombrement de TCP.

Une réponse étant reçue, la demande correspondante se déplace de la fenêtre d'envoi vers la fenêtre de réponse achevée, et la taille de AvailableSendWindowSize augmente de 1 si le Max_Send_Window_Size n'a pas été atteint. Si une temporisation se produit en attendant une réponse, la perte ou l'encombrement de message est indiqué(e). Si cela se produit, alors:

- Aucun message complémentaire ne doit être placé dans l'OutstandingList tant que l'OutstandingList n'a pas d'abord été vide.
- La CurrentSendWindowLimit doit être mise à un (1).
- Les messages qui étaient dans l'OutstandingList au moment de l'effondrement doivent faire l'objet de répétitions de tentative dans l'ordre, selon les politiques de répétitions de tentative définies ci-dessus. Les répétitions de tentative utilisent le repli exponentiel si la première tentative ne réussit pas. Les répétitions de tentative doivent continuer jusqu'à ce qu'une réponse soit reçue ou jusqu'à ce que le nombre maximal de répétitions de tentative ait été atteint. Lorsque l'une ou l'autre de ces conditions se produit, le traitement de messages est considéré comme achevé, et le message doit être retiré de l'OutstandingList.

Les demandes de client peuvent continuer, accumulant le CurrentSendWindowLimit à la valeur de Max_Send_Window_Size en utilisant la procédure décrite ci-dessus.

EXEMPLE dans un exemple de la façon dont les fenêtres sont utilisées dans une situation où il n'y a aucune répétition de tentative:

Soit Max_Send_Window_Size constant, égal au nombre maximal de demandes simultanément en cours permis par le contrat. Cette limite est établie par le gestionnaire de système.

Soit CurrentSendWindowSize la variable qui représente le nombre de demandes simultanément en cours qui existent pour le contrat à l'instant t. CurrentSendWindowSize est un nombre entier non négatif inférieur ou égal à Max_Send_Window_Size.

Soit UsedSendWindowSize la variable qui représente le nombre de demandes simultanément en attente qui attendent toujours des réponses.

AvailableSendWindowSize = CurrentSendWindowSize – UsedSendWindowSize.

Hypothèse: Max_Send_Window_Size = 3

T1: initialisation: CurrentSendWindowSize = 1; UsedSendWindowSize = 0; AvailableSendWindowSize = 1;

T2: message M1 envoyé: CurrentSendWindowSize = 1; UsedSendWindowSize = 1; AvailableSendWindowSize = 0;

T3: message M1, réponse reçue: CurrentSendWindowSize = 1; UsedSendWindowSize = 0; AvailableSendWindowSize = 1;

T4: message M2 envoyé: CurrentSendWindowSize = 1; UsedSendWindowSize = 1; AvailableSendWindowSize = 0;

T5: message M3, réponse reçue: CurrentSendWindowSize = 2; UsedSendWindowSize = 0; AvailableSendWindowSize = 2;

Le CurrentSendWindowSize a été incrémenté de 1, car:

a) il a été à la taille 1, et 2 transactions se sont achevées avec succès;

b) CurrentSendWindowSize < Max_Send_Window_Size.

T6: message M4 envoyé: CurrentSendWindowSize = 2; UsedSendWindowSize = 1; AvailableSendWindowSize = 1;

T7: message M5 envoyé: CurrentSendWindowSize = 2; UsedSendWindowSize = 2; AvailableSendWindowSize = 0;

T8 Message M4 et M5, réponses reçues: CurrentSendWindowSize = 2; UsedSendWindowSize = 0; AvailableSendWindowSize = 2;

T9: message M6 envoyé: CurrentSendWindowSize = 2; UsedSendWindowSize = 1; AvailableSendWindowSize = 1;

T10: message M7 envoyé: CurrentSendWindowSize = 2; UsedSendWindowSize = 2; AvailableSendWindowSize = 0;

T11: message M6, réponse reçue: CurrentSendWindowSize = 3; UsedSendWindowSize = 1; AvailableSendWindowSize = 2;

Le CurrentSendWindowSize a été incrémenté de 1, car:

a) il a été à la taille 2, et 3 transactions se sont achevées avec succès;

b) CurrentSendWindowSize < Max_Send_Window_Size.

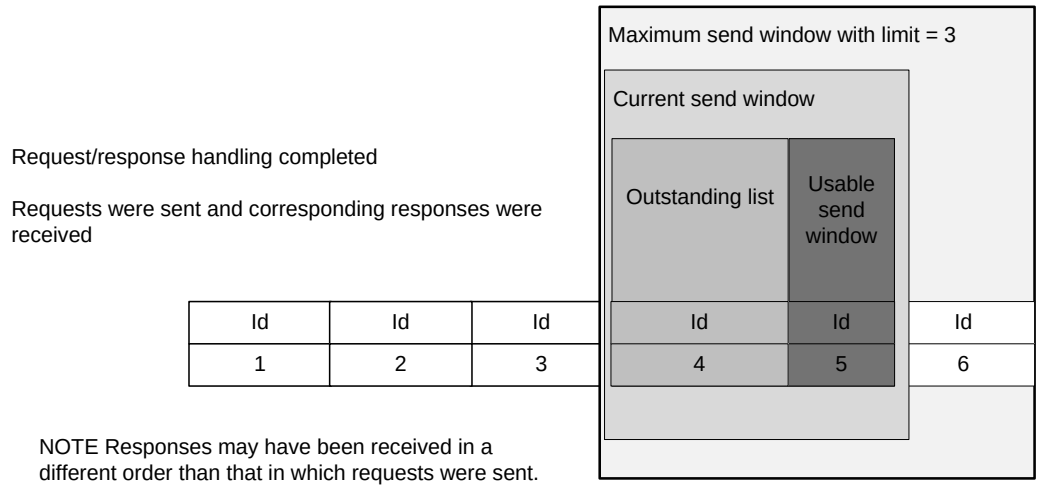
T12: message M8 envoyé: CurrentSend WindowSize = 3; UsedSendWindowSize = 2; Available SendWindowSize = 1;

T13: message M9 envoyé: CurrentSendWindowSize = 3; UsedSendWindowSize = 3; AvailableSendWindowSize = 0;

T14: message M7, réponse reçue: CurrentSendWindowSize = 3; UsedSendWindowSize = 2; AvailableSendWindowSize = 1;

T15: message M10 envoyé: CurrentSendWindowSize = 3; UsedSendWindowSize = 3; AvailableSendWindowSize = 0;

La Figure 116 décrit une situation dans laquelle la fenêtre d'envoi courante ne s'est pas encore accumulée jusqu'à la taille limite maximale de fenêtre d'envoi. Dans cet exemple, la taille limite maximale de fenêtre d'envoi est de trois messages, un message dans la liste en cours a été envoyé et attend une réponse, et un message peut être envoyé avant que la limite de fenêtre d'envoi utilisable ne soit atteinte.



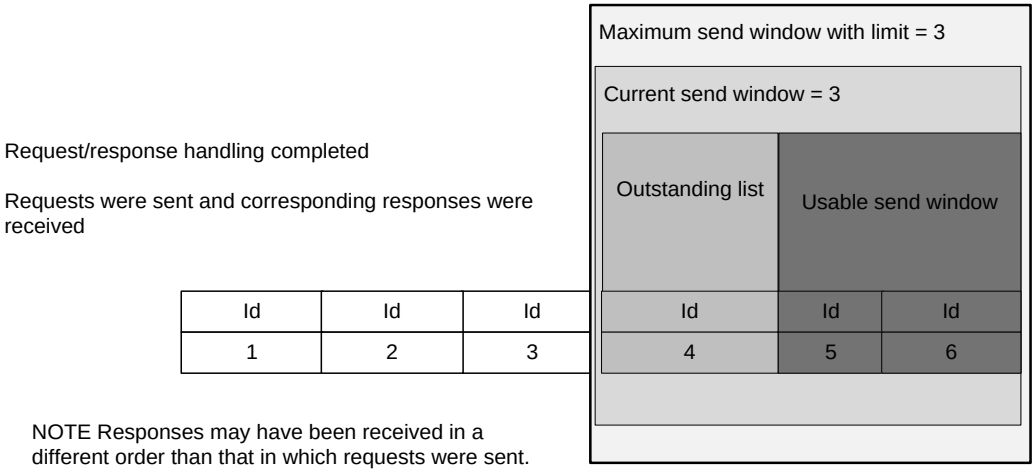
Légende

Anglais	Français
Request/response handling completed	Traitement achevé des demandes/réponses
Requests were sent and corresponding responses were received	Les demandes ont été envoyées et les réponses correspondantes ont été reçues
Maximum send window with limit = 3	Fenêtre d'envoi maximale avec limite = 3
Current send window	Fenêtre d'envoi courante
Outstanding list	Liste en cours
Usable send window	Fenêtre d'envoi utilisable

Anglais	Français
NOTE Responses may have been received in a different order than that in which requests were sent.	NOTE Les réponses peuvent avoir été reçues dans un ordre différent de celui dans lequel les demandes ont été envoyées.

Figure 116 – Exemple 1 de fenêtre d'envoi avec fenêtre d'envoi courante plus petite que la fenêtre d'envoi maximale

La Figure 117 décrit une situation dans laquelle la fenêtre d'envoi courante s'est accumulée jusqu'à la taille limite maximale de fenêtre d'envoi. Dans cet exemple, la taille limite maximale de fenêtre d'envoi est de trois messages, un message dans la liste en cours a été envoyé et attend une réponse, et deux messages peuvent être envoyés avant que la limite de fenêtre d'envoi utilisable ne soit atteinte.

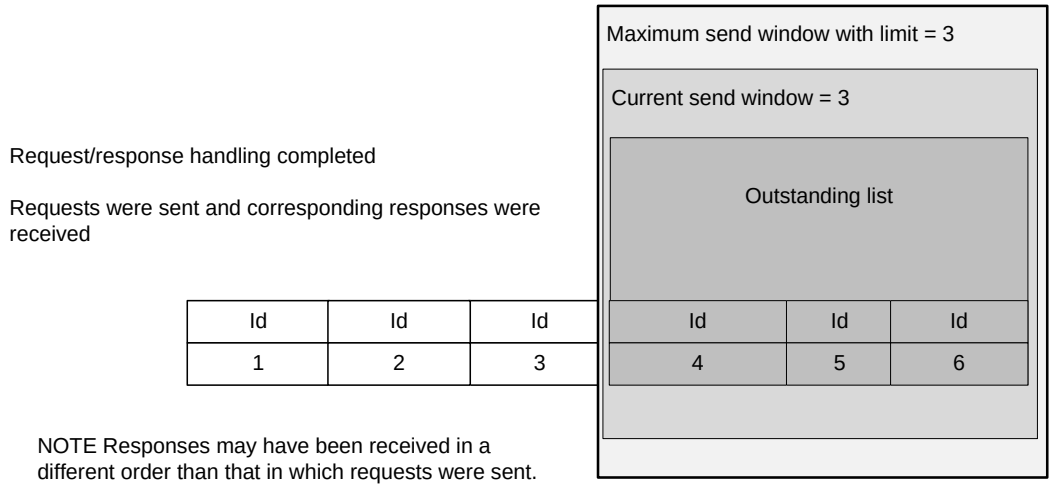


Légende

Anglais	Français
Request/response handling completed	Traitement achevé des demandes/réponses
Requests were sent and corresponding responses were received	Les demandes ont été envoyées et les réponses correspondantes ont été reçues
Maximum send window with limit = 3	Fenêtre d'envoi maximale avec limite = 3
Current send window	Fenêtre d'envoi courante
Outstanding list	Liste en cours
Usable send window	Fenêtre d'envoi utilisable
NOTE Responses may have been received in a different order than that in which requests were sent.	NOTE Les réponses peuvent avoir été reçues dans un ordre différent de celui dans lequel les demandes ont été envoyées.

Figure 117 – Exemple 2 de fenêtre d'envoi avec fenêtre d'envoi courante de la même taille que la fenêtre d'envoi maximale et largeur de fenêtre d'envoi utilisable non nulle

La Figure 118 décrit une situation dans laquelle la fenêtre d'envoi courante s'est accumulée jusqu'à la taille limite maximale de fenêtre d'envoi. Dans cet exemple, la fenêtre d'envoi maximale est de trois messages, et trois messages ont été envoyés et attendent des réponses



Légende

Anglais	Français
Request/response handling completed	Traitement achevé des demandes/réponses
Requests were sent and corresponding responses were received	Les demandes ont été envoyées et les réponses correspondantes ont été reçues
Maximum send window with limit = 3	Fenêtre d'envoi maximale avec limite = 3
Current send window = 3	Fenêtre d'envoi courante = 3
Outstanding list	Liste en cours
NOTE Responses may have been received in a different order than that in which requests were sent.	NOTE Les réponses peuvent avoir été reçues dans un ordre différent de celui dans lequel les demandes ont été envoyées.

Figure 118 – Exemple 3 de fenêtre d'envoi, avec fenêtre d'envoi courante de la même taille que la fenêtre d'envoi maximale et largeur de fenêtre d'envoi utilisable de zéro

12.12.4.2.4 Recherche d'encombrement par sondage

Certaines configurations de système sont plus susceptibles que d'autres d'encourir une perte de message en raison d'un encombrement de réseau. Dans des configurations de système où l'encombrement est plus vraisemblable, une application peut souhaiter réguler ses demandes de service d'AL selon que l'encombrement de réseau est présent ou absent. Pour ce faire, une application peut rechercher l'encombrement par sondage. Pour effectuer un tel sondage, l'application peut s'engager dans un simple échange d'un seul message.

NOTE 1 Le sondage est prévu pour les besoins de diagnostic seulement. Un seul message est utilisé pour vérifier que les sondes ne surchargent pas le réseau et pour s'assurer que la réponse à une sonde est discernable.

La demande de message à utiliser lors du sondage doit être un service lecture non concaténé. La demande de lecture et la réponse de lecture correspondante pour la sonde doivent chacune s'inscrire au sein d'un seul fragment de DL. N'importe quel attribut d'objet peut être utilisé comme sonde; cependant, il convient (mais cela n'est pas exigé) que le même objet et le même attribut soient utilisés de façon cohérente pour le sondage. Par exemple, un attribut normalisé de l'UAPMO peut être utilisé pour sonder des UAP, et un attribut normalisé du DMO de DMAP pour sonder le DMAP car ces objets doivent être présents dans les applications correspondantes.

Si la temporisation de la sonde n'expire pas avant la réception de la réponse, alors il convient que l'application suppose qu'il n'y a aucun encombrement. Cependant, si la réponse n'est pas retournée avant que l'intervalle de temporisation des répétitions de tentative ne se soit écoulé, cela indique une plus forte probabilité de présence de l'encombrement de réseau. Dans cette situation, le processus d'application doit autoréguler ses activités de communication en mettant son CurrentWindowSize à 1. Voir 12.12.4.2.3. Si l'application souhaite envoyer un autre message de sondage d'encombrement, elle peut le faire en

utilisant le repli exponentiel tel que décrit en 12.12.4.2.2.1, mais elle doit utiliser un identificateur de demande temporellement discernable pour chaque sonde de message.

NOTE 2 Une telle distinction permet à une application de calculer le RTT spécifique au sondage d'encombrement et de prendre des décisions relatives à l'encombrement en conséquence en fonction des données de RTT relatives au sondage d'encombrement.

Par exemple, un UAP dans l'appareil X peut produire une demande de service lecture pour l'attribut d'état normalisé exigé de l'UAPMO exigé contenu au sein de l'application de destination dans l'appareil Y. Cette demande de lecture peut être traitée comme sonde d'encombrement initiée par un processus d'application.

Dans le cas spécifique d'un téléchargement descendant ou d'un téléchargement montant, une application peut rechercher l'encombrement par sondage. Dans ces situations, le sondage d'encombrement doit être accompli comme suit:

- Pour sonder avant de débuter une opération de téléchargement descendant, la sonde de client doit être une demande de lecture à l'attribut MaxDownloadSize de l'objet UploadDownload.
- Pour rechercher l'encombrement par sondage pendant une opération de téléchargement descendant, la sonde de client doit être une demande de service lecture de l'attribut de LastBlockDownloaded de l'objet UploadDownload.
- Pour la recherche par sondage avant de débuter une opération de téléchargement montant, la sonde de client doit être une demande de lecture à l'attribut MaxUploadSize de l'objet UploadDownload.
- Pour rechercher l'encombrement par sondage pendant une opération de téléchargement montant, la sonde de client doit être une demande de service lecture à l'attribut LastBlockUploaded de l'objet UploadDownload.

12.12.5 Contrat de service de communication

Un UAP fait une demande de contact au DMAP local par l'intermédiaire d'un SAP d'UAPME-n afin d'établir un accord pour un service de communication dont l'UAP a besoin. Si le besoin peut être satisfait, le DMAP fournit un identificateur de contrat de service à l'UAP qui représente l'accord. L'identificateur de contrat est passé de l'UAP à l'ASL lorsqu'il fait une demande de service d'ASL. Cet ID de contrat de service est alors utilisé par la suite de protocoles de communication inférieure pour identifier les caractéristiques spécifiques à une couche du contrat qui ont été établies dans les couches inférieures de suite de protocoles de communication par le DMAP local comme étant une partie intégrante de l'établissement du contrat de service. La communication d'informations exigées, de l'UAP vers le DMAP, afin d'acquérir un identificateur de contrat de service est une question interne à un appareil et, donc, n'est pas spécifiée par la présente norme.

Tous les contrats de communication ont un jeu d'informations de base. Les informations exigées complémentaires dépendent du type de relations de communication désirées. Par exemple, une relation éditer/s'abonner (P/S) pour la communication périodique exige des spécifications de la phase et de la période désirées.

La présente norme ne spécifie pas comment déterminer les informations dont l'UAP a besoin pour spécifier les caractéristiques d'un contrat. Par exemple, de telles informations peuvent être configurées, telles que la périodicité et la phase à utiliser dans une communication programmée, ou de telles informations peuvent être déterminées par le fournisseur de l'appareil qui contient l'UAP.

Les demandes de contrat peuvent être négociées à la baisse par le gestionnaire de système. Les politiques d'UAP concernant le traitement des contrats négociés à la baisse, ainsi que les politiques concernant le traitement d'un contrat décliné, ne relèvent pas du domaine d'application de la présente norme. Voir 6.3.11 pour des informations complémentaires relatives aux informations qui ont besoin d'être spécifiées pour demander un contrat.

La période d'édition est représentée par une valeur entière signée de 16 bits. Une valeur positive indique la période de publication sous la forme d'un multiple de 1 s (par exemple: une valeur de 5 spécifie une période d'édition de 5 s; une valeur de 3 600 indique une période de publication de 1 h). Une valeur négative indique une publication sur une fraction de 1 s (par exemple: -4 indique une publication tous les ¼ s, -2 indique une publication toutes les ½ s). Une valeur nulle indique qu'il convient qu'il ne se produise aucune publication.

Il convient que la périodicité sélectionnée soit basée sur l'efficacité de l'opération avec cette pratique de processus normalisée et typique.

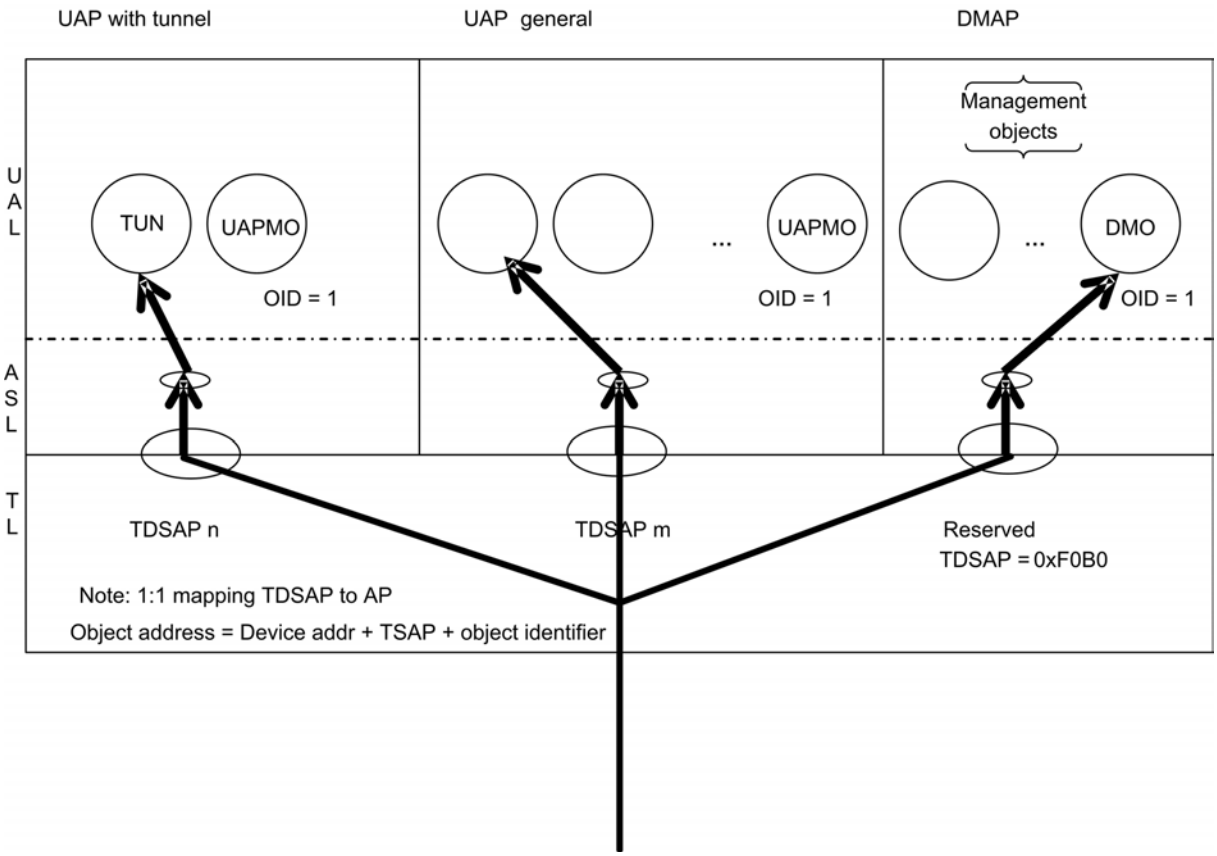
La connaissance par le DMAP du port de TSAP de destination n'est pas une exigence pour créer un contrat de service, car l'établissement de contrat est concerné par des ressources pour la communication sur des conditions de réseau, tandis que le port de destination est utilisé dans l'appareil de destination, après que la communication sur le réseau s'est produite.

NOTE Les politiques de répéter la tentative d'établissement d'un contact en cas d'échec d'établissement de contrat ou de révocation d'un contrat sont des comportements de l'appareil dans son ensemble (par opposition aux comportements d'un composant au sein de l'appareil). Les comportements au niveau d'un appareil sont débattus en 6.3.11.2.4.2.

12.13 Adressage d'AL

12.13.1 Généralités

Certaines informations sont exigées pour adresser un objet, un attribut d'objet, un élément d'attribut d'objet (par exemple, un élément d'une structure ou un élément d'une matrice), ou une méthode d'objet dans des communications natives. La Figure 119 représente le modèle d'adressage général pour les objets de processus d'UAL.



Légende

Anglais	Français
UAP with tunnel	UAP avec tunnel

Anglais	Français
UAP general	UAP général
Management objects	Objets de gestion
Reserved TDSAP=0xF0B0	Réservé TDSAP=0xF0B0
Note:1:1mapping TDSAP to AP	Note: Mapping biunivoque (1:1) de TDSAP à AP
Object address = Device addr + TSAP + object identifier	Adresse d'objet = addr appareil + TSAP + identificateur d'objet

Figure 119 – Modèle d'adressage général

12.13.2 Adressage d'objet

Un objet est adressé dans les communications en monodiffusion en spécifiant:

- son adresse physique d'appareil conteneur;
- le TDSAP de DL qui est utilisé pour communiquer avec le processus UAL unique qui contient l'objet (c'est-à-dire, le TDSAP est en correspondance 1:1 à un processus d'application);
- un numéro de port T qui correspond à un TDSAP particulier; et
- l'identificateur d'objet unique au sein du processus d'UAL.

Des ports T doivent être assignés dans l'ordre consécutif à des TDSAP, en commençant par le premier port T disponible. Par exemple, le TDSAP numéro 0 doit être corrélé avec le premier port T 0xF0B0. Le TDSAP numéro 1 doit être corrélé avec le deuxième port, 0xF0B1, et ainsi de suite.

Des TDSAP particuliers, et leurs ports T correspondants, sont réservés par la présente norme afin d'être bien connus par toutes les applications. Spécifiquement, le DMAP dans chaque appareil doit avoir le numéro de port de transport 0xF0B0 réservé, qui est associé au TDSAP numéro 0. Le SMAP dans un appareil doit avoir le numéro de port de transport 0xF0B1 réservé, qui est associé au TDSAP numéro 1. Les appareils qui n'ont pas un SMAP présent ne doivent pas utiliser le numéro de port T de 0xF0B1.

Il convient que les UAP prévus pour un grand nombre de messages utilisent les ports T de 0xF0B2 à 0xF0BF, car ils sont représentés sous forme compressée sur le réseau, minimisant ainsi l'utilisation des ressources réseau, ainsi que l'encombrement de RF et la demande en énergie de l'appareil. Voir 11.4.4 pour plus de détails relatifs aux TDSAP et aux ports T.

Afin de réduire au maximum le codage des messages d'application, il convient que les identificateurs d'objet soient alloués consécutivement, en commençant à 1.

NOTE L'identificateur d'objet 0 est réservé par la présente norme à l'usage de l'objet de gestion de processus d'application contenu au sein de tous les processus d'application.

La communication en multidiffusion n'est pas prise en charge.

12.13.3 Adressage d'attribut d'objet

Un attribut d'un objet est adressé en spécifiant:

- l'adressage de son objet conteneur; et
- l'identificateur d'attribut d'objet au sein de l'objet.

Afin de réduire au maximum le codage des attributs de messages d'application, il convient que les attributs soient alloués consécutivement, en commençant à 1.

NOTE L'identificateur d'attribut 0 est réservé par la présente norme comme moyen de référence pour toute une somme.

12.13.4 Adressage d'attribut d'objet

12.13.4.1 Généralités

L'adressage pour des attributs est défini en fonction du type d'attribut. La présente norme prend en charge les types d'attribut suivants:

a) types scalaires normalisés définis par la présente norme;

b) matrices d'éléments homogènes ou hétérogènes à une seule dimension (unidimensionnelles) d'origine 1 du type a);

c) matrices d'éléments à deux dimensions (bidimensionnelles) d'origine [1,1], où la première dimension indexe une matrice homogène du type b).

Les structures de données normalisées définies par la présente norme sont modélisées et accédées sous la forme de matrices d'éléments hétérogènes unidimensionnelles d'origine 1. Ainsi l'accès au $k^{\text{ème}}$ membre d'une structure de données, tel qu'il est énuméré dans l'ordre de déclaration de ses éléments membres, est fourni en accédant à son $k^{\text{ème}}$ élément comme si la structure de données était une matrice hétérogène d'origine 1.

NOTE En termes de programmation, cela signifie que le $k^{\text{ème}}$ membre de la structure s, qui pourrait être référencé en programmation sous la forme s.memberName$_k$, est accédé comme s'il était $s[k]$, où k représente l'index ordinal d'origine 1 du membre dans la déclaration contenue.

Les éléments d'une matrice bidimensionnelle qui constituent eux-mêmes des structures ou des matrices sont accessibles uniquement en représentant ces éléments individuels sous la forme de chaînes d'octets de taille uniforme.

12.13.4.2 Scalaires

La présente norme prend en charge l'accès à des attributs qui sont scalaires et se présentent sous forme des types suivants:

- Boolean, mappé à Boolean8 ou à Boolean1 lorsqu'il se trouve dans une structure de données condensées;
- Integer, mappé à Integer8, Integer16, Integer32, Unsigned8, Unsigned16, Unsigned32, Unsigned64, Unsigned128, ou à UnsignedN où $N < 16$ lorsqu'il se trouve dans une structure de données condensées;
- Float, mappé à Float32 ou Float64;
- VisibleString, mappé à VisibleStringN lorsque N est fixe ou déterminé par le contexte;
- OctetString, mappé à OctetStringN lorsque N est fixe ou déterminé par le contexte;
- BitString, mappé à BitStringN lorsque N est fixe ou déterminé par le contexte;
- SymmetricKey, mappé à OctetString16 dans cette édition de la présente norme.

NOTE 1 Chaque BitString est représenté sous forme d'un nombre entier d'octets, ou sous forme d'un nombre approprié de bits adjacents lorsqu'il se trouve dans une structure de données condensées.

NOTE 2 Voir 12.22.3 sur les types de données pour les types scalaires pris en charge par la présente norme.

NOTE 3 OctetString et BitString fournissent un moyen pour l'acheminement transparent des informations inintelligibles pour la couche de protocole d'acheminement.

12.13.4.3 Adresses de protocole structuré traitées comme scalaires

Les adresses suivantes sont également considérées comme scalaires lorsqu'elles sont utilisées dans les structures de données de la présente norme, même si leurs propres normes de définition spécifient une sous-structure pour l'élément:

- IPv6Address, mappée à Unsigned128 pour la prise en charge d'une comparaison numérique simple;

NOTE 1 La sous-structure de cette classe d'adresse est spécifiée dans l'IETF RFC 2460 et dans ses normes associées.

- EUI64Address, mappée à Unsigned64 pour la prise en charge d'une comparaison numérique simple;

NOTE 2 La sous-structure de cette classe d'adresse est spécifiée dans les lignes directrices Guidelines for 64-bit Global Identifier (EUI-64™) de l'IEEE.

- DL16Address, mappée à Unsigned16 pour la prise en charge d'une comparaison numérique simple;

Dans l'IEEE 802.15.4, la valeur 0xFFFF est l'adresse DL16Address de diffusion, alors que toute valeur comprise dans la plage 0x0000..0x7FFD peut être assignée à une DLE sous forme d'adresse DL16Address en monodiffusion. Cependant, la présente norme réserve les valeurs 0 pour indiquer une adresse DL16Address non assignée. Ainsi, pour la présente norme, la plage des adresses DL16Addresses en monodiffusion est 0x0001..0x7FFF.

NOTE 3 L'IEEE 802.15.4 réserve la valeur 0xFFFE. L'IETF RFC4944 (6LoWPAN sur l'IEEE 802.15.4) spécifie que la plage 0x80FF..0x9FFF est réservée pour les sous-réseaux D locaux en multidiffusion. Le Paragraphe 9.1.6.4 spécifie que la plage 0xA000..0x0AFFF est réservée par la présente norme pour les numéros de graphes utilisés dans les chemins de source.

12.13.4.4 Matrices unidimensionnelles et structures de données normalisées

La présente norme prend en charge l'accès aux structures de données normalisées et aux matrices, qu'elles soient d'éléments scalaires ou de structures de données normalisées.

L'accès pris en charge à une matrice ou à une structure de données normalisée qui n'est pas contenue au sein d'une matrice se présente comme suit:

- une matrice unidimensionnelle ou une structure normalisée a peut être accessible en sa totalité en spécifiant l'accès au membre zéro (par exemple, une valeur "index" de 0, a[0]);
- un membre unique d'une matrice unidimensionnelle ou d'une structure normalisée a peut être accessible en identifiant l'index d'origine 1 du membre k souhaité, comme spécifié en 12.13.4.1;
- un membre scalaire k d'un membre de structure normalisée b d'une structure normalisée a (par exemple, $a.b.k$, où a et b sont les structures normalisées et k est une grandeur scalaire prise en charge par la présente norme);
- un élément de matrice unidimensionnelle b d'une structure normalisée a peut être accessible en totalité en spécifiant l'accès au membre zéro (par exemple, $a.b$[0], où a est une structure normalisée et b est une matrice unidimensionnelle, et la matrice b est composée soit de scalaires soit de structures normalisées, tels que définis par la présente norme);
- un élément simple d'une matrice unidimensionnelle b de structures normalisées qui est un membre d'une structure normalisée a (par exemple, $a.b[k]$, où a est une structure normalisée, b est une matrice unidimensionnelle composée soit de scalaires soit de structures normalisées, tels que définis par la présente norme, et k est l'index d'origine 1 du membre d'intérêt).

12.13.4.5 Matrices unidimensionnelles

La présente norme prend en charge l'accès à une matrice unidimensionnelle, dont les membres individuels sont des scalaires ou des structures de données normalisées telles que définies par la présente norme, comme suit:

- un élément simple d'une matrice unidimensionnelle constituée de scalaires (par exemple, $a[k]$, où a spécifie la matrice et k spécifie l'élément dans la matrice);
- une matrice unidimensionnelle de scalaires ou de structures de données normalisées peut être accessible en sa totalité (par exemple, a[0], où a spécifie la matrice, et 0 spécifie l'accès à la totalité de la matrice);

- un élément d'une matrice unidimensionnelle constituée de structures normalisées (par exemple, $a[k][0]$, où a spécifie la matrice, k spécifie l'élément dans la matrice qui est la structure normalisée, et 0 spécifie l'accès à la structure du membre entier);
- un membre scalaire d'une structure normalisée contenue dans une matrice unidimensionnelle (par exemple, $a[k].j$, où a spécifie la structure telle que définie par la présente norme, k spécifie l'élément dans la matrice qui est la structure normalisée, et j spécifie le membre au sein de la structure normalisée);
- une matrice unidimensionnelle contenue comme membre d'une matrice unidimensionnelle (par exemple, $a[k][0]$, où a est une matrice de structures normalisées, k spécifie un élément de la matrice, et 0 spécifie l'accès à la totalité de la matrice);
- un membre d'une matrice unidimensionnelle de scalaires ou de structures normalisées contenu comme membre d'une matrice unidimensionnelle (par exemple, $a[k][j]$, où a spécifie la matrice de portée extérieure, k spécifie un élément de cette matrice qui est lui-même une matrice, et j représente l'élément de la matrice de portée intérieure).

12.13.4.6 Matrices bidimensionnelles

La présente norme prend en charge l'accès à une matrice bidimensionnelle, composée d'une matrice homogène unidimensionnelle de matrices de scalaires homogènes ou hétérogènes unidimensionnelles, telles que définies par la présente norme, qui se présentent comme suit:

a) un élément scalaire d'une matrice bidimensionnelle (par exemple, $a[k][j]$);

b) une matrice bidimensionnelle en sa totalité (par exemple, $a[0][0]$);

c) une ligne d'une matrice bidimensionnelle (par exemple, $a[k][0]$); ou

d) une colonne d'une matrice bidimensionnelle (par exemple, $a[0][k]$).

NOTE La forme d'adressage d) spécifie une *tranche* de la matrice, où le résultat est une matrice unidimensionnelle dont les éléments sont le $k^{ème}$ membre de chaque sous-matrice. Ce mode d'accès par tranche permet un accès sélectif à tout membre (élément) de chaque structure de données de composant figurant dans une matrice de structures de données construites à l'identique.

12.13.5 Adressage de méthode d'objet

Une méthode d'objet est adressée en spécifiant:

- l'adressage de son objet conteneur; et
- l'index unique d'objet de l'identificateur de méthode d'objet.

12.14 Objets de gestion

Des objets de gestion normalisés pour gérer l'appareil comme un tout sont définis dans la présente norme. Ces objets sont définis en 6.2 et sont accessibles par un MP contenu dans une UAL qui peut inclure, par exemple, un objet de gestion pour prendre en charge l'identification de l'appareil, des objets de gestion pour chaque couche de la suite de protocoles de communication, et un objet de gestion pour rapporter des alertes issues de l'appareil.

NOTE Bien que chaque objet suive à la trace ses propres conditions d'événement et d'alarme, la production de rapports relatifs à de telles conditions est spécifiée par un seul ARMO pour l'appareil dans son ensemble. Cet objet gère les aspects comprenant, mais sans s'y limiter, la(les) file(s) d'attente de rapports d'alertes locale(s), le(s) temporisateur(s) local(locaux) associé(s) à la réémission si un acquittement d'alerte individuelle n'est pas reçu, le traitement de débordement de files d'attente d'alertes locales, et des demandes de récupération d'alarme. Voir 6.2.7.2 pour plus de détails.

12.15 Objets utilisateurs

12.15.1 Généralités

Les objets normalisés pouvant être contenus dans un UAP sont définis pour permettre l'interfonctionnabilité à travers les industries et les segments. Ces objets peuvent être

indépendants vis-à-vis de toute industrie (à savoir, applicables à travers les industries prises en charge par la présente norme), ou être dépendants d'une industrie (c'est-à-dire, applicables à une industrie prise en charge par la présente norme, mais non utilisés par les industries).

12.15.2 Objets indépendants vis-à-vis de toute industrie

12.15.2.1 Généralités

Les objets normalisés (UAPMO, ARO, UDO, Concentrator, Dispersion, Tunnel, et Interface) sont applicables à travers les industries prises en charge par la présente norme.

12.15.2.2 Objet de gestion d'UAP

12.15.2.2.1 Généralités

Il y a exactement un objet de gestion de UAP (UAPMO) adressable pour chaque UAP pris en charge par l'AL définie par la présente norme. L'identificateur d'objet numérique d'un objet indique une instance d'objet particulière. L'identificateur d'objet numérique de l'UAPMO dans chaque UAP doit être fixe et doit avoir la valeur un (1). Cet objet facilite la gestion commune des processus d'application au sein d'un appareil. Des attributs de cet objet sont utilisés pour spécifier des informations telles que la version/révision du processus d'application et le statut logique du processus d'application. Par exemple, un attribut de l'UAPMO indique si l'UAP correspondant est actif ou inactif.

NOTE 1 Un UAPMO peut prendre en charge la gestion d'un groupe (jeu) particulier d'objets au sein de l'UAP.

NOTE 2 L'instanciation dynamique des UAP ne relève pas du domaine d'application de la présente norme.

12.15.2.2.2 Attributs d'objet

Un UAPMO a les attributs définis dans le Tableau 240.

Tableau 240 – Attributs d'objet de gestion d'UAP *(1 de 2)*

Nom du type d'objet normalisé: UAP management object (UAPMO, objet de gestion d'UAP)				
Identificateur du type d'objet normalisé: 1				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
ObjectIdentifier	Identificateur-clé d'objet	Identificateur unique pour l'objet	Type: Unsigned16 Classification: Constant	N/A
UAP_ID	Identificateur-clé d'objet	SAP de TLDE associé	Type: Initiative locale (telle que définie par la TL locale) Classification: Constant	SAP de TDLE local
UAP_TL_Port	Identificateur-clé d'objet	Port T associé	Type: Unsigned16 Classification: Constant	NOTE 1 La spécification de l'UAP par rapport à sa TL locale relève d'une initiative locale. NOTE 2 Le transport définit le jeu de valeurs hexadécimales 0xF0B0+*n* (où *n* peut se situer dans la plage 0..15) pour spécifier la représentation la plus comprimée (en 4 bits) pour la communication
Réservé pour une utilisation future	0	-	-	-
VersionRevision	1	VersionRevision de l'UAP	Type: VisibleString Taille maximale: 64 octets Classification: Constant Accessibilité: Lecture seule	Identification lisible par l'homme associée à l'objet de gestion d'UAP NOTE Le fournisseur d'UAP détermine le contenu de cet attribut
Etat	2	Statut de l'UAP	Type: Unsigned8 Classification: Static Accessibilité: Lecture seule Valeur par défaut: 1 Valeurs nommées: 0: inactif; 1: actif; 2: en panne.	Voir Tableau 241
Commande	3	Commande pour changer l'état de l'UAP	Type: Unsigned8 Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: 0 Valeurs nommées: 0: aucune; 1: arrêt; 2: démarrage; 3: réinitialisation de logiciel; 4: réinitialisation de matériel	La valeur "none" ne doit pas être indiquée dans une demande d'écriture. Une réinitialisation de logiciel doit préserver les données de configuration/mise en service. Une réinitialisation de matériel renvoie l'application à ses valeurs de réglage d'usine par défaut

Tableau 240 *(2 de 2)*

Nom du type d'objet normalisé: UAP management object (UAPMO, objet de gestion d'UAP)				
Identificateur du type d'objet normalisé: 1				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
MaxRetries	4	Nombre maximal de répétitions de tentative de demande client que le processus d'application enverra afin d'avoir une communication C/S réussie	Type: Unsigned8 Classification: Static Accessibilité: Lecture seule Valeur par défaut: 3 Plage valide: 0..8 Classification: Static Accessibilité: Lecture seule	Le nombre de répétitions de tentative envoyées pour un message particulier peut varier par message sur la base de la détermination du processus applicatif de l'importance du message. Par exemple, certains messages ne peuvent pas du tout faire l'objet de répétitions de tentative, et d'autres peuvent être répétés le nombre maximal de fois
Nombre d'objets dans l'UAP y compris cet UAPMO	8	Nombre d'objets dans l'UAP y compris cet UAPMO	Type: Unsigned8 Classification: Static Accessibilité: Lecture seule Valeur par défaut: 1 Plage valide: > 0	Tous les UAP doivent avoir un UAPMO, par conséquent la valeur par défaut est indiquée comme étant égale à 1 (un). La valeur réelle de cet attribut doit être le nombre total d'objets contenus dans l'UAP, y compris l'UAPMO
Array of UAP contained objects	9	Identification des objets et du type contenus dans cet UAP	Type: Array of ObjectIDandType Classification: Static Accessibilité: Lecture seule	Voir Tableau 271
Static_Revision_ Level	10	Niveau de révision des données statiques associées à tous les objets de gestion	Type: Unsigned16 Classification: Static Accessibilité: Lecture seule Valeur par défaut: 0	Le niveau de révision est incrémenté chaque fois qu'une valeur d'attribut statique de n'importe quel objet contenu dans cet UAP change
Réservé pour usage futur par la présente norme	5..7 11..63	-	-	*N* octets de contenu présentement indéfini

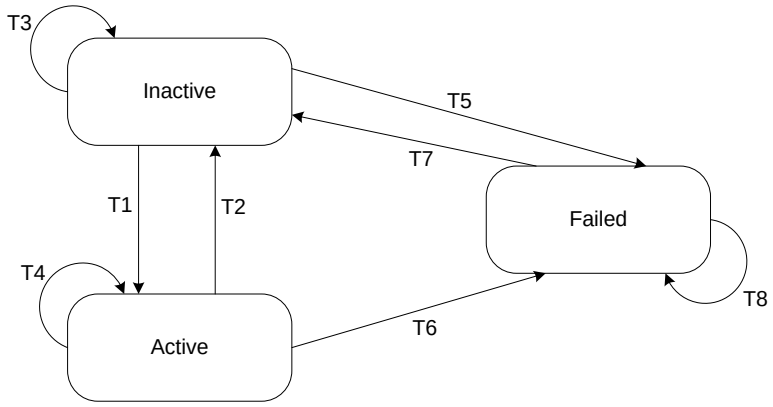
12.15.2.2.3 Table d'états pour l'objet de gestion d'UAP

Le Tableau 241 décrit la table d'états pour l'objet de gestion d'UAP.

Tableau 241 – Table d'états pour l'objet de gestion d'UAP

Transition	Etat actuel	Evénement(s)	Action(s)	Etat suivant
T1	Inactive	Write(Command, Start)	Write.rsp(success)	Active
T2	Active	Write(Command,Stop)	Write.rsp(success)	Inactive
T3	Inactive	Write(Command,Stop)	Write.rsp(success)	Inactive
		Write(toute commande Reset)	Write.rsp(operationAccepted)	
T4	Active	Write(Command,Stop)	Write.rsp(success)	Active
		Write(toute autre commande)	Write.rsp(objectStateConflict)	
T5	Inactive	Write(Command,Stop)	Write.rsp(failed) Note – Echec de démarrage	Failed
T6	Active	Problème d'application	N/A	Failed
T7	Failed	Write(toute commande Reset)	Write.rsp(operationAccepted)	Inactive
T8	Failed	Write(toute commande autre que Reset)	Write.rsp(objectStateConflict)	Failed

La Figure 120 montre le diagramme d'états de l'objet de gestion d'UAP.



Légende

Anglais	Français
Inactive	Inactif
Active	Actif
Failed	Echoué

Figure 120 – Diagramme d'états de l'objet de gestion d'UAP

12.15.2.2.4 Méthodes d'objets normalisés

Un objet de gestion d'UAP a des méthodes telles que définies dans le Tableau 242.

Tableau 242 – Méthodes d'objet de gestion d'UAP

Nom du type d'objet normalisé: UAP management object (objet de gestion d'UAP)		
Identificateur du type d'objet normalisé: 1		
Nom de méthode	**ID de méthode**	**Description de la méthode**
Null	0	Réservé par la présente norme pour usage futur
Réservé pour usage futur par la présente norme	0..127	Ces identificateurs de méthode sont réservés pour usage futur par la présente norme
Implementation-specific use	128..255	Ces identificateurs de méthode sont disponibles pour une utilisation spécifique à une mise en œuvre

12.15.2.3 Objet récepteur d'alerte

12.15.2.3.1 Généralités

Il peut y avoir jusqu'à quatre objets récepteurs d'alerte dans un appareil, un par catégorie de rapports d'alerte. Ces objets récepteurs d'alerte peuvent recevoir plus d'une catégorie de rapports d'alerte. Les catégories des rapports d'alerte reçues par les objets d'alerte doivent être uniques; c'est-à-dire, si un objet récepteur d'alerte reçoit des alertes de catégorie X issues de l'ASL, aucun autre objet d'alerte dans l'appareil ne peut également recevoir des alertes de catégorie X issues de l'ASL. Ces objets récepteurs d'alerte peuvent être contenus dans les mêmes processus ou des processus différents (par exemple, un objet récepteur d'alerte pour des alertes de sécurité peut être contenu dans un processus d'application, alors qu'un autre pour des alertes de processus peut être contenu dans un autre processus d'application).

NOTE La séparation d'alertes, ou une consolidation et une production à nouveau de rapports d'alertes, s'il y a lieu, relève d'une initiative locale de processus d'application, ne relevant pas du domaine d'application de la spécification d'AL.

12.15.2.3.2 Attributs d'objet

Un objet récepteur d'alerte peut recevoir des alertes issues d'une ou plusieurs sources de rapports d'alerte. L'objet a les attributs définis dans le Tableau 243.

Tableau 243 – Attributs d'objet récepteur d'alerte

Nom du type d'objet normalisé: Alert-receiving object (objet récepteur d'alerte)				
Identificateur du type d'objet normalisé: 2				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
ObjectIdentifier	Identificateur-clé d'objet	Identificateur unique pour l'objet	Type: Unsigned16 Classification: Constant Plage valide: > 0	N/A
Réservée pour un usage futur	0	-	-	-
Catégories	1	BitString de catégories d'alerte indiquant quelle instance d'objet prend en charge la réception	Type: BitString Classification: Static Accessibilité: Lecture seule Valeur par défaut: 0 Indices nommés: 0: alertes d'appareil; 1: alertes de communication; 2: alertes de sécurité; 3: alertes de processus; 4..7: réservés pour un usage futur par la présente norme	N/A
Erreurs	2	Compte de rapports reçus pour une catégorie que l'objet de réception indiqué ne prend pas en charge	Type: Unsigned16 Classification: Dynamic Accessibilité: Lecture seule Valeur par défaut: 0	Revient à 0 lorsque la valeur maximale est atteinte
Réservé pour usage futur par la présente norme	3..63	-	-	*N* octets de contenu présentement indéfini

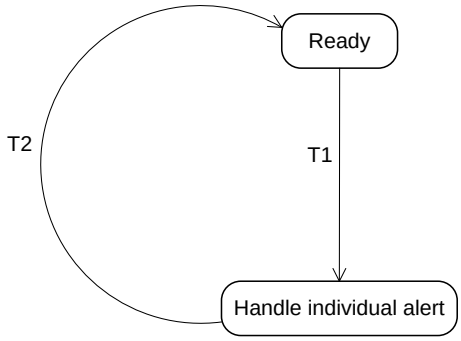
12.15.2.3.3 Table d'états pour le traitement d'AlertReport

Le Tableau 244 indique les états pour traiter la réception d'un AlertReport.

Tableau 244 – Table d'états pour traiter une réception d'AlertReport

Transition	Etat actuel	Evénement(s)	Action(s)	Etat suivant
T1	Prêt	AlertReport.ind reçue	Noter le rapport d'alerte de processus issu de l'appareil X	Traiter l'alerte individuelle dans le rapport
T2	Traiter l'alerte individuelle dans le rapport	Vérifier la catégorie	Catégorie valide: Acquitter le rapport d'alerte et le traiter	Prêt
			Catégorie non valide: Incrémenter la valeur des instances de Alert-Receiving Object de son attribut Errors	

La Figure 121 montre le diagramme d'états pour la réception d'alertes.

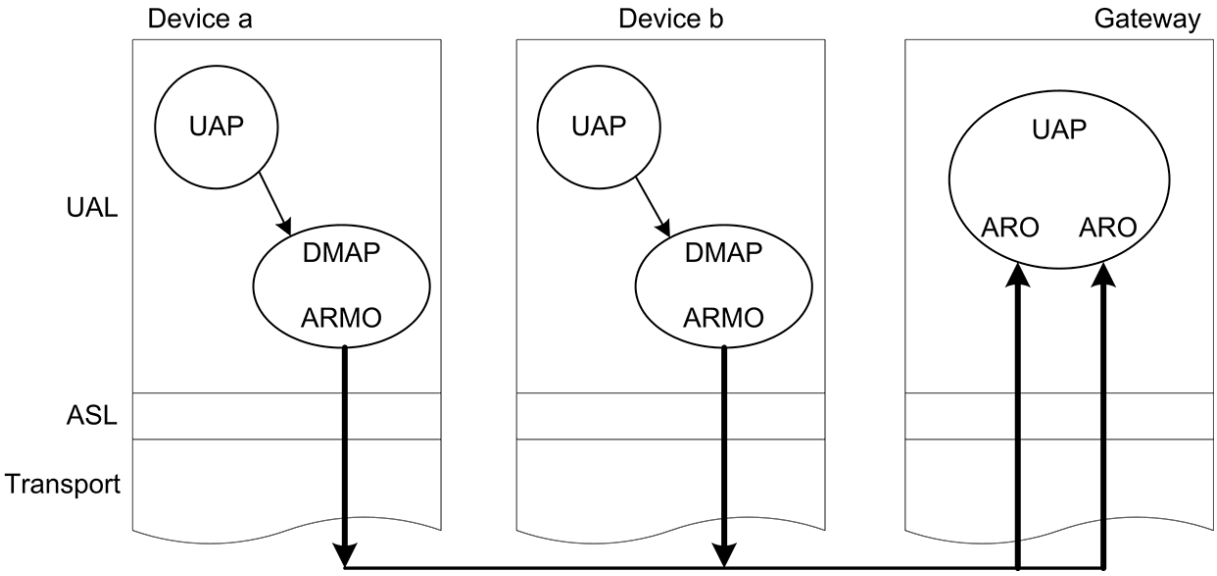


Légende

Anglais	Français
Ready	Prêt
Handle individual alert	Traiter une alerte individuelle

Figure 121 – Diagramme d'états de réception de rapports d'alertes

La Figure 122 montre un exemple de production de rapports issus de plusieurs appareils sources vers plusieurs objets récepteurs d'alertes contenus dans un seul UAP d'un seul appareil puits.



Légende

Anglais	Français
Device a	Appareil a
Device b	Appareil b
Gateway	Passerelle
Transport	Transport

Figure 122 – Exemple de rapports d'alerte

12.15.2.3.4 Méthodes d'objets normalisés

Un objet AlertReceiving a les méthodes définies dans le Tableau 245.

Tableau 245 – Méthodes d'objet AlertReceiving

Nom du type d'objet normalisé: Object AlertReceiving		
Identificateur du type d'objet normalisé: 2		
Nom de méthode	**ID de méthode**	**Description de la méthode**
Null	0	Réservé par la présente norme pour usage futur
Réservé pour usage futur par la présente norme	0..127	Ces identificateurs de méthode sont réservés pour usage futur par la présente norme
Implementation-specific use	128..255	Ces identificateurs de méthode sont disponibles pour une utilisation spécifique à une mise en œuvre

12.15.2.4 Objet UploadDownload

12.15.2.4.1 Généralités

Un objet UploadDownload est utilisé soit pour télécharger en montant, soit pour télécharger en descendant, des informations vers un appareil. L'objet UploadDownload peut être utilisé pour prendre en charge des opérations telles que télécharger en descendant une nouvelle version de firmware d'exploitation ou télécharger en descendant le nouveau code contenu dans un UAP ou des données en masse exigées par l'UAP. L'objet UploadDownload maintient des informations de contrôle de révision pour spécifier ce qui a été téléchargé en descendant et/ou ce qui est disponible pour téléchargement montant.

Un objet UploadDownload est susceptible de prendre en charge le téléchargement montant ou le téléchargement descendant pour un seul jeu sémantique d'informations. Un objet UploadDownload doit prendre en charge une seule opération de téléchargement montant ou de téléchargement descendant à la fois.

Un processus peut avoir zéro ou plus d'instances de l'objet UploadDownload. Plusieurs instances de l'objet UploadDownload sont exigées si plus d'un jeu sémantique d'informations est nécessaire pour télécharger en montant ou télécharger en descendant son contenu exigé.

NOTE 1 L'effet local d'un téléchargement montant ou d'un téléchargement descendant de processus d'application (par exemple, la création de nouveaux objets visibles du réseau comme résultat d'un téléchargement descendant) relève d'une initiative locale, en dehors du domaine d'application de la présente norme.

NOTE 2 Les objets UploadDownload sont utilisables pour prendre en charge des opérations telles que le téléchargement montant d'informations statistiques ou historiques issues de l'appareil en vue de leur analyse. Un objet UploadDownload est utilisable pour mettre à jour le logiciel/firmware dans l'appareil cible.

NOTE 3 La prise en charge d'un téléchargement descendant en multidiffusion est un sujet de future normalisation. Pour prendre en charge des charges en multidiffusion à un ensemble spécifique d'appareils, il est actuellement envisagé d'utiliser un outil de configuration pour configurer les relations adresse de multidiffusion/appareil/objet pour les objets dans l'ensemble de multidiffusion.

12.15.2.4.2 Attributs d'objet

Un objet UploadDownload a les attributs définis dans le Tableau 246. Des attributs sont inclus dans ce type d'objet afin de fournir des conseils de temporisation des communications à un niveau application au client qui est en communication avec l'objet UploadDownload.

NOTE Les conseils supplémentaires au client, tels que ceux relatifs à la syntonisation de la temporisation des communications (par exemple, relatifs aux retards de communication réseau dus à la topologie du graphe de messagerie traversé, aux potentiels retards de mise en file d'attente, etc.), utilisables pour syntoniser le comportement d'application client, sont transparents au processus d'application, et par conséquent l'application elle-même ne peut pas fournir des conseils complets.

Tableau 246 – Attributs d'objet UploadDownload *(1 de 4)*

Nom du type d'objet normalisé: Objet UploadDownload				
Identificateur du type d'objet normalisé: 3				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
ObjectIdentifier	Identificateur-clé d'objet	Identificateur unique pour l'objet	Type: Unsigned16 Classification: Constant Plage valide: > 0	N/A
Réservée pour un usage futur	0	-	-	-
OperationsSupported	1	Indique si cet objet prend en charge les téléchargements montants et/ou les téléchargements descendants	Type: Unsigned8 Classification: Constant Accessibilité: Lecture seule Valeurs nommées: 0: uniquement le téléchargement montant en monodiffusion de la taille définie; 1: uniquement le téléchargement descendant en monodiffusion de la taille définie; 2: téléchargement montant en monodiffusion et téléchargement descendant en monodiffusion de la taille définie; 3..15: réservés pour usage futur par la présente norme	N/A
Description	2	Identification lisible par l'homme du contenu associé	Type: VisibleString TAILLE (0..64) Classification: Static Accessibilité: Lecture seule	
Etat	3	Etat de l'instance de l'objet UploadDownload	Type: Unsigned8 Classification: Dynamic Accessibilité: Lecture seule Valeur par défaut: 0 Valeurs nommées: 0: inactive; 1: en téléchargement descendant; 2: en téléchargement montant; 3: en application; 4: DLComplete; 5: ULComplete; 6: DLError; 7: ULError	Voir le tableau d'états ci-dessous
Commande	4	Commande d'action à cet objet	Type: Unsigned8 Classification: Non-cacheable Accessibilité: Lecture/écriture Valeurs nommées: 0: réinitialisation; 1: application (utilisée pour Download uniquement); 2..15: réservés pour usage futur par la présente norme	Voir Tableau 254

Tableau 246 *(2 de 4)*

Nom du type d'objet normalisé: Objet UploadDownload				
Identificateur du type d'objet normalisé: 3				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
MaxBlockSize	5	Taille maximale d'un bloc qui peut être accepté pour un téléchargement descendant ou fourni pour un téléchargement montant	Type: Unsigned16 Classification: Static Accessibilité: Lecture seule Valeur par défaut: 1 à (MaxNPDUsize – Max TL header size – max(sizeof (additional coding of AL UploadData service request), additional coding of sizeof(AL DownloadData service response)) Plage valide: 0..taille maximale pour les données dans une APDU	Unité: octets La valeur ne doit pas excéder la quantité maximum de données qui peuvent être acheminées dans une seule APDU conformément au contrat de communication. En outre, l'espace dans l'APDU doit être laissé pour le codage relatif à un service. Les tailles de bloc acheminées peuvent être plus petites que cette valeur, mais elles ne doivent pas être plus grandes
MaxDownloadSize	6	Taille maximale disponible pour le téléchargement descendant dans son ensemble	Type: Unsigned32 Classification: Static Accessibilité: Lecture seule Valeur par défaut: 0	Unité: octets
MaxUploadSize	7	Taille disponible pour le téléchargement montant	Type: Unsigned32 Classification: Static Accessibilité: Lecture seule Valeur par défaut: 0	Unité: octets
DownloadPrepTime	8	Temps exigé, en secondes, pour se préparer à un téléchargement descendant	Type: Unsigned16 Classification: Static Accessibilité: Lecture seule Valeur par défaut: 0	Temps exigé entre le moment de l'envoi de la réponse StartDownload et le moment où l'objet peut traiter un DownloadData
DownloadActivationTime	9	Temps en secondes pour que l'objet applique un contenu nouvellement téléchargé en descendant	Type: Unsigned16 Classification: Static Accessibilité: Lecture seule Valeur par défaut: 0	N/A
UploadPrepTime	10	Temps exigé, en secondes, pour se préparer à un téléchargement montant	Type: Unsigned16 Classification: Static Accessibilité: Lecture seule Valeur par défaut: 0	Temps exigé entre le moment de l'envoi de la réponse StartUpload et le moment où l'objet peut accepter un UploadData

Tableau 246 *(3 de 4)*

Nom du type d'objet normalisé: Objet UploadDownload				
Identificateur du type d'objet normalisé: 3				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
UploadProcessingTime	11	Temps type en secondes pour que cet objet d'application traite une demande de télécharger en montant un bloc	Type: Unsigned16 Classification: Static Accessibilité: Lecture seule Valeur par défaut: 0	Ces informations sont prévues pour permettre à un client d'une opération Upload de syntoniser sa messagerie relative au téléchargement montant de façon à correspondre au fonctionnement de l'instance d'objet UploadDownload particulière. Par exemple, un client peut utiliser ce temps pour aider à déterminer sa politique de temporisations/répétions de tentative, ou à déterminer quand invoquer une méthode sur l'instance d'objet
DownloadProcessingTime	12	Temps type en secondes pour que cet objet d'application traite un bloc téléchargé en descendant	Type: Unsigned16 Classification: Static Accessibilité: Lecture seule Valeur par défaut: 0	Ces informations peuvent être utilisées par un client d'une opération Download pour syntoniser sa messagerie relative au téléchargement descendant de façon à correspondre au fonctionnement de l'instance d'objet UploadDownload particulière. Par exemple, un client peut utiliser ce temps pour aider à déterminer sa politique de temporisations/répétions de tentative, ou à déterminer quand invoquer une méthode sur l'instance d'objet
CutoverTime	13	Temps (en secondes) spécifié pour appliquer le contenu de téléchargement descendant	Type: TAINetworkTime Classification: Static Accessibilité: Lecture / Ecriture Valeur par défaut initiale: 0	Le contenu téléchargé en descendant s'appliquera au temps spécifié par cet attribut
LastBlockDownloaded	14	Numéro du dernier bloc téléchargé en descendant avec succès	Type: Unsigned16 Classification: Static Accessibilité: Lecture seule Valeur par défaut: 0	Mis à jour lorsqu'une réponse d'exécution à une méthode DownloadData est retournée. Le comptage des numéros de bloc doit commencer à 1 (un). Voir 12.15.2.4.5.3

Tableau 246 *(4 de 4)*

Nom du type d'objet normalisé: Objet UploadDownload				
Identificateur du type d'objet normalisé: 3				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
LastBlockUploaded	15	Numéro du dernier bloc téléchargé en montant avec succès	Type: Unsigned16 Classification: Static Accessibilité: Lecture seule Valeur par défaut: 0	Mis à jour lorsqu'une réponse d'exécution à une méthode UploadData est retournée. Le comptage des numéros de bloc doit commencer à 1 (un). Voir 12.15.2.4.5.3
ErrorCode	16	Erreur Upload ou Download	Type: Unsigned8 Classification: Static Accessibilité: Lecture seule Valeur par défaut: 0 Valeurs nommées: 0: noError; 1: timeout; 2: clientAbort; 18: InconsistentContent; 27: InsufficientDevice Resources; 3..17, 19..26, 28..63: réservés pour usage futur par la présente norme; 64..255: manufacturerSpecific	Mis à jour lorsqu'il se produit une erreur en téléchargeant en montant ou en téléchargeant en descendant vers cet objet. L'erreur est éliminée lorsque l'objet passe de l'état d'erreur à l'état inactif. Utiliser InconsistentContent pour indiquer que l'appareil n'a pas basculé comme programmé, en raison d'un problème avec la charge utile de téléchargement descendant. Utiliser InsufficientDevice Ressources pour indiquer que le téléchargement descendant n'a pas pu être achevé par manque de mémoire ou d'autres ressources
Réservé pour usage futur par la présente norme	16..63	-	-	-
La présente norme ne propose pas de politique de gestion de cycle de vie de produit ou de contrôle des versions de produit. La description peut être utilisée pour indiquer l'interchangeabilité des versions, ou pour identifier des caractéristiques/réparations/moutures de logiciel. La valeur maximale de tout codage complémentaire pour le codage d'application selon la présente norme est 9 octets. Les réalisateurs peuvent souhaiter consulter l'IETF RFC 2348 concernant les recommandations pour une taille maximale des PDU.				

12.15.2.4.3 Méthodes d'objets normalisés

L'initiation d'un transfert de données en masse en téléchargement montant ou téléchargement descendant exige d'abord de conclure un accord entre les objets d'application correspondants pour participer au transfert de données.

Toute coordination complémentaire exigée pour assurer la disponibilité de l'appareil répondeur à accepter une demande téléchargement montant ou téléchargement descendant relève de la responsabilité du processus d'UAL qui démarre une opération de transfert en masse.

La messagerie client/serveur pour accéder à des informations de coordination issues d'un attribut ou d'un ensemble d'attributs de l'objet UploadDownload peut être utilisée pour prendre en charge cette activité de coordination. Plus précisément, une demande de lecture peut être utilisée bien avant de démarrer le transfert en masse afin qu'un client recueille les informations relatives à la communication du transfert en masse qui sont spécifiques à une instance de l'objet UploadDownload. Une fois que l'accord est conclu, l'application client commande le moment où les données sont fournies (pour un téléchargement descendant) à l'objet UploadDownload, ou demandées (pour un téléchargement montant) en provenance de l'objet UploadDownload. Lorsque le transfert est achevé, le client indique que le transfert est terminé. En outre, le client peut fermer le transfert s'il détermine qu'il convient de ne pas mener le transfert de données entier à son terme.

L'application serveur peut demander que le transfert de données soit abandonné si elle détermine que le transfert de données ne peut pas être mené à son terme ou qu'il convient de ne pas le mener à son terme.

Il convient qu'une application serveur prenant une décision d'abandon basée sur un manque de communication issue du client permette au moins la prise en charge des répétitions normalisées de tentative par défaut et la politique de temporisation des répétitions de tentative pour la politique de communication client/serveur afin d'établir une temporisation appropriée.

Tout comme avec d'autres communications d'application, de la largeur de bande d'émission doit être allouée par un contrat de service de communication afin de prendre en charge un transfert de données en masse. La largeur de bande pour le transfert de données en masse n'est pas considérée comme étant une largeur de bande dédiée telle qu'utilisée pour la messagerie périodique, mais elle est plutôt considérée comme étant une largeur de bande partagée telle qu'utilisée pour la messagerie apériodique. L'utilisation de la largeur de bande partagée parmi tous les utilisateurs de largeur de bande partagée par un appareil est dépendante d'une combinaison de priorité de contrat global et de priorité de message. La priorité de contrat est définie par le gestionnaire de système. La priorité de message est définie par le processus d'application.

NOTE 1 Toute coordination ou ordonnancement exigé(e) de plusieurs images vers différents objets UploadDownload relève de la responsabilité du processus d'application hôte. Les différentes images chargeables ou téléchargeables nécessitent des instances d'objet UploadDownload distinctes.

NOTE 2 La sémantique et la syntaxe du contenu et de l'utilisation des informations téléchargées en montant ou téléchargées en descendant ne relèvent pas du domaine d'application de la présente norme. L'activité résultante dans le processus d'application de l'appareil fournissant des données de téléchargement montant ou acceptant des données de téléchargement descendant, autre que la mise à jour de l'objet UploadDownload lui-même, relève d'une initiative locale et, donc, ne relève pas du domaine d'application de la présente norme.

NOTE 3 L'application proxy au sein de l'appareil est unidirectionnelle pour qu'un seul appareil traite le téléchargement descendant plusieurs fois.

Le téléchargement montant à partir d'un seul appareil ou le téléchargement descendant vers un seul appareil utilise un protocole en monodiffusion; c'est-à-dire, le contenu de téléchargement montant ou de téléchargement descendant est envoyé en provenance/à destination d'un seul objet UploadDownload au sein d'un seul appareil.

NOTE 4 Les conventions relatives au contenu et/ou à la désignation des fichiers, si elles sont applicables à un téléchargement montant ou à un téléchargement descendant ne relèvent pas du domaine d'application de la présente norme.

Un objet UploadDownload a les méthodes définies dans le Tableau 247.

Tableau 247 – Méthodes d'objet UploadDownload

Nom du type d'objet normalisé: Objet UploadDownload		
Identificateur du type d'objet normalisé: 3		
Nom de méthode	**ID de méthode**	**Description de la méthode**
Null	0	Réservé par la présente norme pour usage futur
StartDownload	1	Cette méthode est utilisée par un client pour conclure un accord avec un objet UploadDownload afin de participer à un téléchargement descendant pour lequel le client fournira les données, un bloc à la fois
DownloadData	2	Cette méthode est utilisée par un client pour fournir des données à un objet UploadDownload pour une opération convenue de téléchargement descendant
EndDownload	3	Cette méthode est utilisée par un client pour mettre fin à une opération de téléchargement descendant soit qui s'est terminée avec succès, soit que le client souhaite abandonner
StartUpload	4	Cette méthode est utilisée par un client pour conclure un accord avec un objet UploadDownload afin de participer à un téléchargement montant pour lequel le client demandera les données, un bloc à la fois
UploadData	5	Cette méthode est utilisée par un client pour demander des données à un objet UploadDownload pour une opération convenue de téléchargement montant
EndUpload	6	Cette méthode est utilisée par un client pour mettre fin à une opération de téléchargement montant soit qui s'est terminée avec succès, soit que le client souhaite abandonner
Réservé pour usage futur par la présente norme	7..127	Ces identificateurs de méthode sont réservés pour usage futur par la présente norme
Implementation-specific use	128..255	Ces identificateurs de méthode sont disponibles pour une utilisation spécifique à une mise en œuvre
L'approche utilisée pour le téléchargement montant et le téléchargement descendant a des racines dans les expériences de plusieurs normes acceptées, y compris, sans y être nécessairement limité, la Fieldbus Foundation, la Device Net International, l'IETF RFC 1350 et l'IETF RFC 2347. Les attributs de l'objet UploadDownload fournissent des informations au niveau application pour aider à la détermination de temporisation par un client et, donc, l'IETF RFC 2349 n'est pas suivie. La répétition de tentative d'acquittement telle que proposée dans l'IETF RFC 2347 n'est pas adoptée pour les raisons suivantes: • les cas d'utilisation pilotant la présente norme, et l'ensemble convenu d'exigences techniques auxquelles la présente norme devait satisfaire, disposent du grand volume de communications éditer/s'abonner (P/S), avec une utilisation très limitée des opérations de téléchargement montant/téléchargement descendant; • les opérations de téléchargement montant/téléchargement descendant qui ont été définies ne sont pas en temps critique; • l'application client reçoit un retour d'informations du serveur si le serveur est en train de récupérer des doublons, et peut choisir de mettre fin à l'opération; • l'application serveur sait quand elle envoie des messages d'erreur en retour au client, et peut choisir d'abandonner l'opération.		

12.15.2.4.4 Méthode StartDownload

12.15.2.4.4.1 Généralités

Le Tableau 248 décrit la méthode StartDownload de l'objet UploadDownload.

Tableau 248 – Méthode StartDownload de l'objet UploadDownload

<table>
<tr><th colspan="5">Nom du type d'objet normalisé: Objet UploadDownload</th></tr>
<tr><th colspan="5">Identificateur du type d'objet normalisé: 3</th></tr>
<tr><th>Nom de méthode</th><th>ID de méthode</th><th colspan="3">Description de la méthode</th></tr>
<tr><td rowspan="11">StartDownload</td><td>1</td><td colspan="3">Un client utilise la méthode StartDownload pour indiquer à une instance de l'objet UploadDownload qu'il désire télécharger en descendant des données à partir de l'objet</td></tr>
<tr><td colspan="4">Arguments d'entrée</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument</td><td>Description de l'argument</td></tr>
<tr><td>1</td><td>BlockSize</td><td>Unsigned16</td><td>La taille d'un bloc de données en octets qui sera téléchargé en descendant</td></tr>
<tr><td>2</td><td>DownloadSize</td><td>Unsigned32</td><td>La taille totale des données devant être téléchargées en descendant, en octets</td></tr>
<tr><td>3</td><td>DownloadMode</td><td>Unsigned8</td><td>Le mode de fonctionnement désiré</td></tr>
<tr><td colspan="4">Arguments de sortie</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument</td><td>Description de l'argument</td></tr>
<tr><td colspan="4">Aucun</td></tr>
</table>

12.15.2.4.4.2 Description de la méthode

Un client utilise la méthode StartDownload pour indiquer à une instance de l'objet UploadDownload qu'il désire télécharger en descendant l'objet, et pour indiquer les paramètres du téléchargement descendant dans la liste d'arguments d'entrée. L'objet UploadDownload peut accepter ou rejeter le téléchargement descendant, en l'indiquant par l'intermédiaire de la liste d'arguments de sortie.

Si un objet UploadDownload accepte de participer à une opération de téléchargement descendant, il ne doit pas accepter une autre opération de téléchargement descendant, ou une opération de téléchargement montant, tant que l'opération de téléchargement descendant en cours n'est pas terminée ou l'objet n'a pas été réinitialisé.

12.15.2.4.4.3 Arguments d'entrée

Les arguments d'entrée comprennent:

- BlockSize, qui indique la taille d'un bloc de données en octets. Tous les blocs doivent avoir la même taille, sauf que le dernier bloc d'un téléchargement descendant peut contenir un plus petit nombre positif d'octets.
- DownloadSize, qui représente la taille totale des données devant être téléchargées en descendant, en octets;
- DownloadMode, qui indique le mode de fonctionnement désiré. La valeur valide pour cet argument indique la monodiffusion et elle est représentée par une valeur de zéro.

12.15.2.4.4.4 Arguments de sortie

Il n'y a aucun argument de sortie pour cette méthode.

12.15.2.4.4.5 Codes de réponse

Les codes suivants de retour d'informations sont valides pour cette méthode:

- operationAccepted;
- invalidBlockSize;
- invalidDownloadSize;
- unexpectedMethodSequence;
- insufficientDeviceResources;
- deviceHardwareCondition; et
- ceux qui sont définis par le fournisseur.

12.15.2.4.5 Méthode DownloadData

12.15.2.4.5.1 Généralités

Le Tableau 249 décrit la méthode DownloadData de l'objet UploadDownload.

Tableau 249 – Méthode DownloadData de l'objet UploadDownload

Nom du type d'objet normalisé: Objet UploadDownload				
Identificateur du type d'objet normalisé: 3				
Nom de méthode	**ID de méthode**	**Description de la méthode**		
DownloadData	2	Un client utilise la méthode DownloadData pour fournir des données à un objet UploadDownload qui a accepté d'être téléchargé en descendant		
	Arguments d'entrée			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument**	**Description de l'argument**
	1	BlockNumber	Unsigned16	BlockNumber téléchargé
	2	Data	OctetString	Les données pour le bloc téléchargé en descendant La taille maximale de cette chaîne peut varier, si bien qu'elle peut être différente pour des objets UploadDownload de destination différents
	Arguments de sortie			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument**	**Description de l'argument**
	1	CurrentBlockNumber	Unsigned16	Cet argument est présent si le serviceFeedbackCode indique soit un bloc hors séquence, soit un doublon

12.15.2.4.5.2 Description de la méthode

StartDownload est utilisé d'abord pour amener l'objet UploadDownload à être d'accord avec le téléchargement descendant.

Des données sont envoyées à un bloc à la fois, séquentiellement du bloc ayant le plus petit numéro au bloc ayant le grand numéro en utilisant la méthode de DownloadData. Une seule invocation de la méthode DownloadData peut être en cours à la fois; par exemple, si la méthode DownloadData pour le bloc n été invoquée, la méthode DownloadData pour le bloc n +1 ne doit pas être invoquée tant qu'une réponse de succès contenant les arguments de sortie pour le téléchargement descendant du bloc n n'a pas été reçue par le client.

L'objet UploadDownload peut indiquer qu'il a besoin d'abandonner par l'intermédiaire de l'argument de sortie MethodStatus.

Si un client d'une opération de téléchargement montant ou de téléchargement descendant produit plusieurs invocations de méthodes de transfert de données pour le même bloc, cela peut être dû soit à un problème relatif au réseau (par exemple, la demande n'atteint pas le serveur), soit à un problème au niveau de l'appareil serveur. Dans cette situation, le client peut utiliser la méthode appropriée d'arrêt d'opération (EndDownload ou EndUpload) pour mettre fin à l'opération.

Si un client d'un téléchargement montant ou d'un téléchargement descendant reçoit des réponses multiples de dataSequenceError, cela peut être dû soit à des problèmes relatifs au réseau (par exemple, perte d'une réponse d'invocation de méthode), soit à des problèmes au niveau de l'appareil serveur. Dans cette situation, le client peut utiliser la méthode appropriée d'arrêt d'opération (EndDownload ou EndUpload) pour mettre fin à l'opération. De même, si un objet UploadDownload a envoyé plusieurs réponses dataSequenceError, il peut inférer qu'il y a soit des problèmes relatifs au réseau, soit des problèmes au niveau de l'appareil client et il peut choisir d'abandonner l'opération. Si un objet d'UploadDownload indique l'abandon d'une opération, et cet abandon est perdu sur le réseau, la réponse envoyée à une méthode de données consécutive (DownloadData ou UploadData) ou à une méthode de terminaison (EndDownload ou EndUpload) indique que l'objet ne participe plus à une opération de téléchargement montant ou de téléchargement descendant avec ce client en envoyant une réponse indiquant "unexpectedMethodSequence".

12.15.2.4.5.3 Arguments d'entrée

Les arguments d'entrée comprennent:

- BlockNumber, qui est le numéro du bloc pour lequel des données sont fournies, où le comptage des numéros de bloc commence à 1 (un); et
- Data, qui représente les données pour le bloc téléchargé en descendant.

12.15.2.4.5.4 Arguments de sortie

Cet argument BlockNumber est présent si le serviceFeedbackCode indique soit un bloc hors séquence, soit un doublon. L'argument indique le dernier BlockNumber reçu. L'intention est de permettre au client de résoudre une erreur de réception de bloc hors séquence ou en doublon sans abandonner l'opération de téléchargement.

12.15.2.4.5.5 Codes de réponse

Les codes suivants de retour d'informations sont valides pour cette méthode:

- success;
- invalidBlockNumber;
- blockDataError (par exemple, mauvaise taille de bloc; problème de contenu);
- unexpectedMethodSequence;
- insufficientDeviceResources;
- deviceHardwareCondition;
- operationAborted;
- dataSequenceError (doublon, par exemple);
- timingViolation; et
- ceux qui sont définis par le fournisseur.

12.15.2.4.6 Méthode EndDownload

12.15.2.4.6.1 Généralités

Le Tableau 250 décrit la méthode EndDownload de l'objet UploadDownload.

Tableau 250 – Méthode EndDownload de l'objet UploadDownload

<table>
<tr><th colspan="6">Nom du type d'objet normalisé: Objet UploadDownload</th></tr>
<tr><th colspan="6">Identificateur du type d'objet normalisé: 3</th></tr>
<tr><th>Nom de méthode</th><th>ID de méthode (non négatif)</th><th colspan="4">Description de la méthode</th></tr>
<tr><td rowspan="8">EndDownload</td><td>3</td><td colspan="4">Un client utilise la méthode EndDownload pour indiquer que le téléchargement descendant se termine</td></tr>
<tr><td colspan="5">Arguments d'entrée</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument</td><td colspan="2">Description de l'argument</td></tr>
<tr><td>1</td><td>Rationale</td><td>Unsigned8</td><td colspan="2">Cet argument indique au client la raison de l'arrêt de l'opération de téléchargement descendant</td></tr>
<tr><td colspan="5">Arguments de sortie</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument</td><td colspan="2">Description de l'argument</td></tr>
<tr><td colspan="5">Aucun</td></tr>
</table>

12.15.2.4.6.2 Description de la méthode

Un client utilise la méthode EndDownload pour indiquer que l'opération de téléchargement descendant se termine. L'arrêt peut se produire, par exemple, si le téléchargement descendant est arrivé à son terme ou si le client a choisi de mettre fin à l'opération de téléchargement descendant.

La méthode EndDownload peut être envoyée à partir d'un client qui est actuellement engagé dans une opération de téléchargement descendant, comme convenu par la méthode StartDownload.

12.15.2.4.6.3 Arguments d'entrée

L'argument Rationale (justification) indique la raison du client de mettre fin à l'opération de téléchargement descendant. La valeur utilisée doit être issue de l'ensemble suivant:

- 0: téléchargement descendant achevé avec succès; ou
- 1: abandon client

12.15.2.4.6.4 Arguments de sortie

Il n'y a aucun argument de sortie pour cette méthode.

12.15.2.4.6.5 Codes de réponse

Les codes suivants de retour d'informations sont valides pour cette méthode:

- success;
- operationIncomplete;
- unexpectedMethodSequence;
- timingViolation; et
- ceux qui sont définis par le fournisseur.

12.15.2.4.7 Méthode StartUpload

12.15.2.4.7.1 Généralités

Le Tableau 251 décrit la méthode StartUpload de l'objet UploadDownload.

Tableau 251 – Méthode StartUpload de l'objet UploadDownload

<table>
<tr><th colspan="5">Nom du type d'objet normalisé: Objet UploadDownload</th></tr>
<tr><th colspan="5">Identificateur du type d'objet normalisé: 3</th></tr>
<tr><th>Nom de méthode</th><th>ID de méthode</th><th colspan="3">Description de la méthode</th></tr>
<tr><td rowspan="8">StartUpload</td><td>4</td><td colspan="3">Un client utilise la méthode StartUpload pour indiquer à une instance de l'objet UploadDownload qu'il désire télécharger en montant des données à partir de l'objet</td></tr>
<tr><td colspan="4">Arguments d'entrée</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument</td><td>Description de l'argument</td></tr>
<tr><td>1</td><td>DownloadMode</td><td>Unsigned8</td><td>Le mode de fonctionnement désiré</td></tr>
<tr><td colspan="4">Arguments de sortie</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument</td><td>Description de l'argument</td></tr>
<tr><td>1</td><td>BlockSize</td><td>Unsigned16</td><td>La taille d'un bloc de données en octets</td></tr>
<tr><td>2</td><td>UploadSize</td><td>Unsigned32</td><td>La taille totale des données devant être téléchargées en montant, en octets</td></tr>
</table>

12.15.2.4.7.2 Description de la méthode

Un client utilise la méthode StartUpload pour indiquer à une instance de l'objet UploadDownload qu'il désire télécharger en montant des données à partir de l'objet. L'objet UploadDownload peut accepter ou rejeter le téléchargement montant, en l'indiquant par l'intermédiaire de la liste d'arguments de sortie.

Si un objet UploadDownload accepte de participer à une opération de téléchargement montant, il ne doit pas accepter une autre opération de téléchargement montant ou une opération de téléchargement descendant tant que l'opération de téléchargement montant en cours n'est pas terminée ou l'objet n'a pas été réinitialisé.

12.15.2.4.7.3 Arguments d'entrée

Les arguments d'entrée comprennent:

- DownloadMode, qui spécifie le mode de fonctionnement désiré. La valeur valide pour cet argument indique la monodiffusion et elle est représentée par une valeur de zéro.

12.15.2.4.7.4 Arguments de sortie

Les arguments de sortie comprennent:

- BlockSize, qui indique la taille d'un bloc de données en octets. Tous les blocs doivent avoir la même taille, sauf que le dernier bloc d'un téléchargement montant peut contenir un plus petit nombre positif d'octets;
- UploadSize, qui indique la taille des données devant être téléchargées en montant, en octets.

12.15.2.4.7.5 Codes de réponse

Les codes suivants de retour d'informations sont valides pour cette méthode:

- success;
- unexpectedMethodSequence;
- insufficientDeviceResources;
- deviceHardwareCondition;
- ceux qui sont définis par le fournisseur.

12.15.2.4.8 Méthode UploadData

12.15.2.4.8.1 Généralités

Le Tableau 252 décrit la méthode UploadData de l'objet UploadDownload.

Tableau 252 – Méthode UploadData de l'objet UploadDownload

<table>
<tr><th colspan="5">Nom du type d'objet normalisé: Objet UploadDownload</th></tr>
<tr><th colspan="5">Identificateur du type d'objet normalisé: 3</th></tr>
<tr><th>Nom de méthode</th><th>ID de méthode</th><th colspan="3">Description de la méthode</th></tr>
<tr><td rowspan="8">UploadData</td><td>5</td><td colspan="3">Un client utilise la méthode UploadData pour acquérir des données issues d'un objet UploadDownload qui a accepté d'être téléchargé en montant</td></tr>
<tr><td colspan="4">Arguments d'entrée</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument</td><td>Description de l'argument</td></tr>
<tr><td>1</td><td>BlockNumber</td><td>Unsigned16</td><td>Le numéro du bloc pour lequel des données sont demandées</td></tr>
<tr><td colspan="4">Arguments de sortie</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument</td><td>Description de l'argument</td></tr>
<tr><td>1</td><td>Data</td><td>OctetString</td><td>Cet argument contient les données pour le bloc demandé. Cet argument est présent si et seulement si le serviceFeedbackCode indique "success".

Sa taille maximale peut varier par instance d'objet UploadDownload téléchargé en montant</td></tr>
</table>

12.15.2.4.8.2 Description de la méthode

Un client utilise la méthode UploadData pour acquérir des données issues d'un objet UploadDownload qui a accepté d'être téléchargé en montant.

Le StartUpload est utilisé d'abord pour amener l'objet UploadDownload à être d'accord avec le téléchargement montant. Des données sont demandées à un bloc à la fois, séquentiellement du bloc ayant le plus petit numéro au bloc ayant le grand numéro. Une seule invocation de la méthode UploadData peut être en cours à la fois. Par exemple, si la méthode UploadData pour le bloc n a été invoquée, la méthode UploadData pour le bloc n + 1 ne doit pas être invoquée tant que la réponse de succès correspondante contenant les arguments de sortie n'a pas été reçue par le client.

L'objet UploadDownload peut indiquer qu'il a besoin d'abandonner par l'intermédiaire d'un argument de sortie.

12.15.2.4.8.3 Arguments d'entrée

L'argument BlockNumber spécifie le numéro du bloc pour lequel des données sont demandées. Le comptage des numéros de bloc doit commencer à 1 (un).

12.15.2.4.8.4 Arguments de sortie

L'argument Data contient les données pour le bloc demandé. Cet argument est présent si et seulement si le serviceFeedbackCode indique "success".

12.15.2.4.8.5 Service feedback codes (codes de retour d'informations relatives à un service)

Les codes suivants de retour d'informations sont valides pour cette méthode:

- success;
- unexpectedMethodSequence;
- insufficientDeviceResources;
- deviceHardwareCondition;
- operationAborted;
- dataSequenceError (par exemple: doublon, numéro de bloc non valide, numéro de bloc inattendu);
- timingViolation; et
- ceux qui sont définis par le fournisseur.

12.15.2.4.9 Méthode EndUpload

12.15.2.4.9.1 Généralités

Le Tableau 253 décrit la méthode EndUpload de l'objet UploadDownload.

Tableau 253 – Méthode EndUpload de l'objet UploadDownload

<table>
<tr><th colspan="5">Nom du type d'objet normalisé: Objet UploadDownload</th></tr>
<tr><th colspan="5">Identificateur du type d'objet normalisé: 3</th></tr>
<tr><th>Nom de méthode</th><th>ID de méthode (non négatif)</th><th colspan="3">Description de la méthode</th></tr>
<tr><td rowspan="8">EndUpload</td><td>6</td><td colspan="3">Un client utilise la méthode EndUpload pour indiquer que l'opération de téléchargement montant se termine</td></tr>
<tr><td colspan="4">Arguments d'entrée</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument</td><td>Description de l'argument</td></tr>
<tr><td>1</td><td>Rationale</td><td>Unsigned8</td><td>Cet argument indique au client la raison de l'arrêt de l'opération de téléchargement montant</td></tr>
<tr><td colspan="4">Arguments de sortie</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument</td><td>Description de l'argument</td></tr>
<tr><td colspan="4">Aucun</td></tr>
</table>

12.15.2.4.9.2 Description de la méthode

Un client utilise la méthode EndUpload pour indiquer que l'opération de téléchargement montant se termine. L'arrêt peut se produire, par exemple, si le téléchargement montant est

arrivé à son terme ou si le client a choisi de mettre fin à l'opération de téléchargement montant.

La méthode EndUpload peut être envoyée à partir d'un client qui est actuellement engagé dans une opération de téléchargement montant, comme convenu par la méthode StartUpload.

12.15.2.4.9.3 Arguments d'entrée

L'argument Rationale indique la raison du client de mettre fin à l'opération de téléchargement montant. La valeur utilisée doit être issue de l'ensemble suivant:

- 0: téléchargement montant achevé avec succès; ou
- 1: abandon client

12.15.2.4.9.4 Arguments de sortie

Il n'y a aucun argument de sortie pour cette méthode.

12.15.2.4.9.5 Service feedback codes (codes de retour d'informations relatives à un service)

Les codes suivants de retour d'informations sont valides pour cette méthode:

- success;
- operationIncomplete;
- unexpectedMethodSequence;
- timingViolation;
- ceux qui sont définis par le fournisseur.

12.15.2.4.10 Table d'états pour le téléchargement descendant

Le Tableau 254 montre la table d'états du téléchargement descendant.

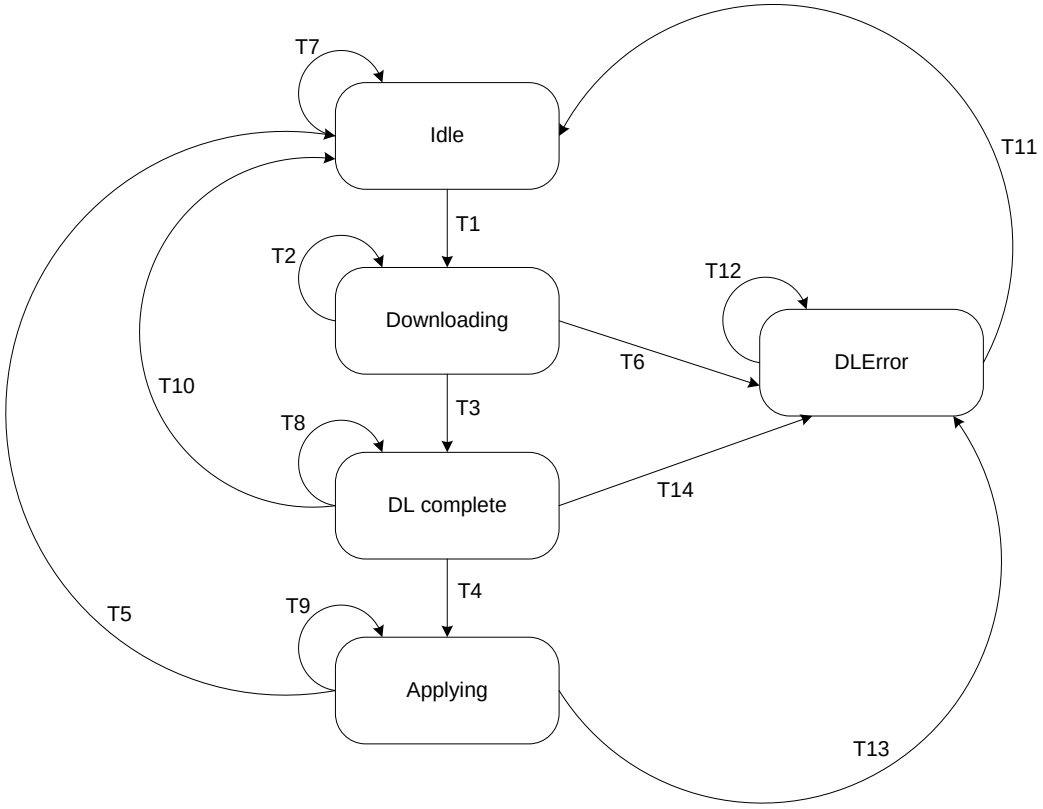
Tableau 254 – Table d'états du téléchargement descendant pour le mode de fonctionnement en monodiffusion *(1 de 2)*

Transition	Etat actuel	Evénement(s)	Action(s)	Etat suivant
T1	Idle	Execute.indicate(StartDownload)	Execute.response(success)	Downloading
T2	Downloading	Execute.indicate(DownloadData) La demande pour le bloc est issue du même objet client qui avait démarré le téléchargement descendant et les paramètres de données de téléchargement descendant sont acceptables	Execute.response(success)	Downloading
		Execute.indicate(StartDownload) ou Execute.indicate(any Upload Method)	Execute.response (objectStateConflict)	
		Execute.indicate(DownloadData) et la demande est issue d'un mauvais client, ou quelque chose est erroné dans les paramètres de données de téléchargement descendant ou dans la temporisation	Execute.response(appropriate error) où l'erreur appropriée peut être, par exemple, invalidArgument, incompatibleMode, timingViolation, etc. NOTE Il relève d'une initiative locale que l'objet UploadDownload détermine si/quand abandonner le téléchargement descendant.	
		Execute.indicate(EndDownload [Success]) et l'objet UploadDownload n'est pas d'accord avec le fait que le téléchargement descendant s'est terminé avec succès	Execute.response(incompatibleMode)	
		Write.indicate(StateCommand.Any value)	Write.response(objectStateConflict)	
T3	Downloading	Execute.indicate(EndDownload [Success])	Execute.response(success)	DLComplete
T4	DLComplete	Write.indicate(StateCommand, Apply)	Write.response(operationAccepted)	Applying (en application)
T5	Applying (en application)	Application réussie	Aucune	Idle
T6	Downloading	Temporiser en attendant une invocation de méthode ultérieure	Mettre à jour l'attribut ErrorCode de l'objet UploadDownload	DLError
		Execute.indicate(EndDownload[Abort])	Mettre à jour l'attribut ErrorCode de l'objet UploadDownload Execute.response (Success)	

Tableau 254 *(2 de 2)*

Transition	Etat actuel	Evénement(s)	Action(s)	Etat suivant
T7	Idle	Execute.indicate(toute méthode Download autre que StartDownload)	Execute.response(objectStateConflict)	Idle
		Execute.indicate(StartDownload) et la demande est inacceptable. Par exemple, un ou plusieurs arguments d'entrée ne sont pas acceptables	Execute.response(erreur appropriée) (invalidObjectID, par exemple)	
		Write.indicate(StateCommand.Any value autre que Reset)	Write.response(objectStateConflict)	
		Write.indicate(StateCommand.Reset)	Write.response(success)	
T8	DLComplete	Execute.indicate(toute méthode Download ou toute méthode Upload)	Execute.response(objectStateConflict)	DL_Complete
T9	Applying	Execute.indicate(toute méthode Download ou toute méthode Upload)	Execute.response(objectStateConflict)	Applying
		Write.indicate(StateCommand.Any value)	Write.response(objectStateConflict)	
T10	DLComplete	Write(StateCommand, Reset)	1. Rejeter le contenu du téléchargement descendant; et 2. Write.rsp(success)	Idle
T11	DLError	Write(StateCommand, Reset)	1. Rejeter le contenu du téléchargement descendant; 2. Effacer l'attribut ErrorCode; et 3. Write.req(success)	Idle
T12	DLError	Toute méthode Upload ou Download	Execute.response (objectStateConflict)	DLError
		Toute commande d'état autre que Write(StateCommand.Reset)	Write.req(objectStateConflict)	
T13	Applying	Echec d'application	Mettre à jour l'attribut ErrorCode de l'objet UploadDownload	DLError
T14	DLComplete	Temporiser en attendant une commande à appliquer	Mettre à jour l'attribut ErrorCode de l'objet UploadDownload	DLError

La Figure 123 montre le diagramme d'états de téléchargement descendant d'un objet Upload/Download.



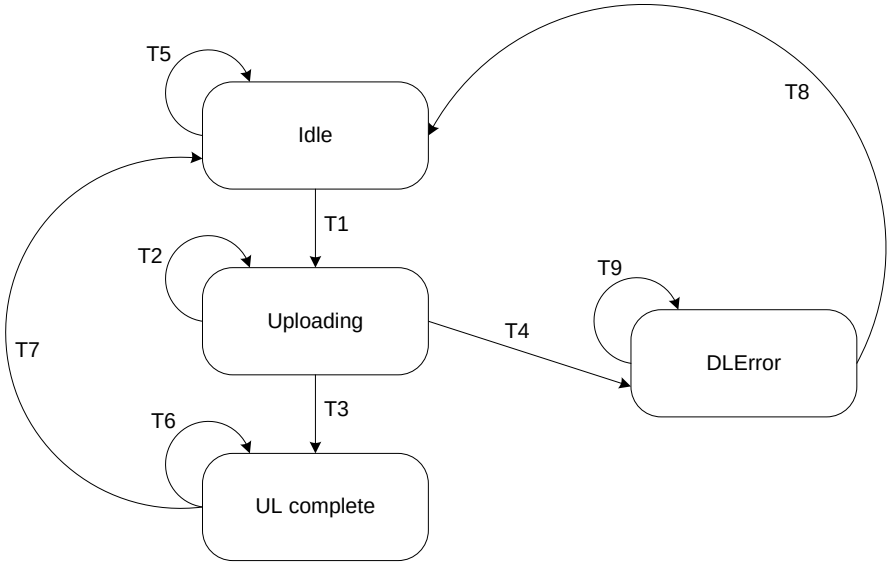
Légende

Anglais	Français
Idle	Inactif
Downloading	En téléchargement descendant
DL complete	DL achevé
Applying	En application
DL Error	Erreur de DL

Figure 123 – Diagramme d'états de téléchargement descendant d'un objet UploadDownload

12.15.2.4.11 Table d'états pour le téléchargement montant

Le Tableau 255 montre la table d'états du téléchargement montant.



Légende

Anglais	Français
Idle	Inactif
Uploading	En téléchargement montant
UL complete	UL achevé
DL Error	Erreur de DL

Figure 124 – Diagramme d'états de téléchargement montant d'un objet UploadDownload

Tableau 255 – Table d'états du téléchargement montant pour le mode de fonctionnement en monodiffusion *(1 de 2)*

Transition	Etat actuel	Evénement(s)	Action(s)	Etat suivant
T1	Idle	Execute.indicate(StartUpload)	Execute.response(Success)	Uploading
T2	Uploading	Execute.indicate(UploadData) La demande pour le bloc est issue du même objet client qui avait démarré le téléchargement montant et les paramètres de données de téléchargement montant sont considérés comme acceptables	Execute.response(Success)	Uploading
		Execute.indicate (StartUpload) ou toute méthode Download	Execute.response (objectStateConflict)	
		Execute.indicate(UploadData) et la demande est issue d'un mauvais client, ou quelque chose est erroné dans les paramètres de données de téléchargement montant ou dans la temporisation	Execute.response(erreur appropriée) où l'erreur appropriée peut être, par exemple, invalidArgument, incompatibleMode, timingViolation, etc. NOTE Il relève d'une initiative locale que l'objet UploadDownload détermine si/quand abandonner le téléchargement montant si cela se produit plus d'une fois consécutivement	
		Execute.indicate(EndUpload [Success]) et l'objet UploadDownload n'est pas d'accord avec le fait que le téléchargement montant s'est terminé avec succès	Execute.response(incompatibleMode)	
		Write.indicate(StateCommand .Any value)	Write.response(objectStateConflict)	
T3	Uploading	Execute.indicate(EndUpload [Success])	Execute.response(Success)	ULComplete
T4	Uploading	Temporiser en attendant une invocation de méthode ultérieure	Mettre à jour l'attribut ErrorCode de l'objet UploadDownload	UL_Error
		Execute.indicate(EndUpload [Success])	1. Mettre à jour l'attribut ErrorCode de l'objet UploadDownload; 2. Execute.response(Success)	

Tableau 255 *(2 de 2)*

Transition	Etat actuel	Evénement(s)	Action(s)	Etat suivant
T5	Idle	Execute.indicate(toute méthode Upload autre que StartUpload)	Execute.response(objectStateConflict)	Idle
		Execute.indicate(StartUpload) et la demande est inacceptable. Par exemple, un ou plusieurs arguments d'entrée ne sont pas acceptables	Execute.response(erreur appropriée) (invalidObjectID, par exemple)	
		Write.indicate(StateCommand. autre que Reset)	Write.response(objectStateConflict)	
		Write.indicate(StateCommand.Reset)	Write.response(success)	
T6	ULComplete	Execute.indicate(toute DownloadMethod) ou toute UploadMethod	Execute.response (objectStateConflict)	UL_Complete
T7	ULComplete	Write(StateCommand, Reset)	Write.req(success)	Idle
T8	ULError	Write(StateCommand, Reset)	1. Effacer l'attribut ErrorCode; et 2. Write.req(success)	Inactif
T9	ULError	Toute méthode Upload ou Download	Execute.response (objectStateConflict)	ULError
		Write(StateCommand.autre que Reset)	Write.req(error)	

La Figure 124 montre le diagramme d'états de téléchargement montant d'un objet UploadDownload.

12.15.2.4.12 Responsabilités du client relatives aux opérations de téléchargement montant/téléchargement descendant

Afin de traiter les retards de message tant dans les demandes que dans les réponses et éviter un effondrement par encombrement dû à une boucle de réémission, seule la première instance d'une réponse indiquant un succès doit amener le prochain bloc de données à être envoyé par l'intermédiaire d'une méthode DownloadData ou à être demandé par l'intermédiaire d'une invocation de méthode UploadData par le client.

NOTE L'intention est d'éviter de recréer des situations historiques telles qu'encourues avec le protocole trivial de transfert de fichier (TFTP), créant le syndrome d'apprenti-sorcier.

12.15.2.5 Objet Concentrator

12.15.2.5.1 Généralités

Un objet Concentrator représente un ensemble des données, recueillies à partir de plusieurs objets dans le même UAP, qui doivent être éditées par un seul service de demande d'édition. Cet objet optimise des messages de publication envoyés depuis un appareil. Plusieurs instances d'objet concentrateur peuvent être utilisées pour représenter plusieurs ensembles des données s'il y a lieu. Une liste d'attributs est fournie pour indiquer les valeurs de données qui sont éditées.

NOTE Le contenu édité représenté par cet objet est établi par configuration. La présente norme ne spécifie pas l'outil de configuration d'appareil.

Un abonné à des données produites par un objet concentrateur doit seulement être un objet dispersion. Il convient que les types de données associés à la liste d'attributs de l'objet

dispersion soient configurés de façon à concorder avec ceux produits par l'objet concentrateur.

Quand un objet concentrateur est configuré par une application hôte, telle qu'une passerelle, l'appareil est responsable d'établir des contrats selon les besoins pour prendre en charge les publications correspondantes. La conception est destinée à prendre en charge deux cas d'utilisation. Dans un cas, l'appareil rejoint le réseau et ensuite, l'hôte configure l'objet concentrateur. Dans l'autre cas, l'objet concentrateur est préconfiguré et l'appareil démarre de façon autonome la publication après qu'il a rejoint le réseau.

Un UAP peut avoir zéro ou plus d'instances de l'objet Concentrator.

12.15.2.5.2 Attributs d'objet

Un objet concentrateur a les attributs définis dans le Tableau 256.

La première fois qu'un UAP reçoit une demande read/write/execute issue d'un point d'extrémité pour lequel il n'a aucun contrat, il doit demander un contrat afin de pouvoir envoyer une réponse de service au point d'extrémité demandeur. L'UAP doit, selon les besoins, retarder la première réponse de service pour accorder du temps pour établir/modifier le contrat.

Tableau 256 – Attributs d'objet Concentrator *(1 de 2)*

Nom du type d'objet normalisé: Objet Concentrator				
Identificateur du type d'objet normalisé: 4				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
ObjectIdentifier	Identificateur-clé d'objet	Identificateur unique pour l'objet	Type: Unsigned16 Classification: Constant Plage valide: > 0	N/A
Réservée pour un usage futur	0	-	-	-
Concentrator ContentRevision	1	Suit un changement apporté à ce qui a été publié; assure que les objets Concentrator (éditeur) et Dispersion (abonné) sont en harmonie	Type: Unsigned8 Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: 0	La révision doit être incrémentée lorsque le complément de données à publier change, à savoir CommunicationEndpoint ou Array of ObjectAttributeIndexAndSize sont modifiés. Attribut inclus dans l'en-tête du Tableau 347
CommunicationEndpoint	2	Sert à identifier l'objet qui reçoit la publication issue de cet objet	Type: Structure de point d'extrémité d'association de communication Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: L'élément valide de point d'extrémité de connexion configuré indique non configuré (c'est-à-dire que le point d'extrémité n'est pas valide)	Ecrire dans cet attribut en dernier lors de la configuration de cet objet; voir Tableau 265
Données de contrat de communication pour la communication programmée	3	Données correspondant au contrat de communication	Type: Communication contract data Classification: Static Accessibilité: Lecture seule	Mis à jour lorsque le contrat correspondant est établi ou résilié; voir Tableau 266
MaximumItems Publishable	4	Nombre maximal d'éléments qui peuvent être édités	Type: Unsigned8 Classification: Constant Accessibilité: Lecture seule Valeur par défaut: Initiative locale	Si cet attribut a une valeur de 0, il indique qu'il n'est pas configuré pour l'édition
NumberItemsPublishing	5	Nombre réel d'éléments publiés	Type: Unsigned8 Classification: Static Accessibilité: Lecture seule Valeur par défaut: 0	Mis à jour lorsque les attributs ObjectAttributeIndexAndSize sont configurés: incrémenté lorsqu'une autre valeur à éditer est ajoutée, et décrémenté lorsqu'une valeur à éditer est enlevée

Tableau 256 *(2 de 2)*

Nom du type d'objet normalisé: Objet Concentrator				
Identificateur du type d'objet normalisé: 4				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
ObjectAttributes	6	Matrice de données pour identifier chaque élément de données publié	Type: Array of Object AttributeIndexAndSize Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: La taille de l'élément est 0	ID d'objet, ID d'attribut, indice d'attribut et taille pour chaque valeur publiée Voir Tableau 264
Réservé pour usage futur par la présente norme	7..63	-	-	-

Les attributs Revision, NumItemsSubscribing et d'ObjAttrIdx peuvent être mis en œuvre dans un appareil afin qu'ils puissent être écrits d'une manière atomique dans une transaction de réseau unique par l'intermédiaire de la concaténation des APDU.

12.15.2.5.3 Méthodes d'objets normalisés

Un objet Concentrator présente les méthodes définies dans le Tableau 257.

Tableau 257 – Méthodes d'objet Concentrator

Nom du type d'objet normalisé: Objet Concentrator		
Identificateur du type d'objet normalisé: 4		
Nom de méthode	**ID de méthode**	**Description de la méthode**
Null	0	Réservé par la présente norme pour usage futur
Réservé pour usage futur par la présente norme	0..127	Ces identificateurs de méthode sont réservés pour usage futur par la présente norme
Implementation-specific use	128..255	Ces identificateurs de méthode sont disponibles pour une utilisation spécifique à une mise en œuvre

12.15.2.6 Objet Dispersion

12.15.2.6.1 Généralités

Un objet dispersion est l'objet abonné correspondant à un objet concentrateur. Cet objet est configuré pour indiquer comment analyser le contenu édité d'un objet concentrateur. Si plusieurs démontages sont exigés, plusieurs objets utilisateurs dispersion doivent être utilisés. Un UAP peut avoir zéro ou plus d'instances de l'objet Dispersion.

NOTE Les objets concentrateur et dispersion sont les objets spéciaux prenant en charge un proxy de publication au sein d'un UAP. Ces objets sont distincts des processus d'application proxy, qui peuvent distribuer de l'information au travers de plusieurs UAP au sein de l'UAL.

12.15.2.6.2 Attributs d'objet

Un objet dispersion a les attributs définis dans le Tableau 258.

Tableau 258 – Attributs d'objet Dispersion *(1 de 2)*

Nom du type d'objet normalisé: Objet Dispersion				
Identificateur du type d'objet normalisé: 5				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
ObjectIdentifier	Identificateur-clé d'objet	Identificateur unique pour l'objet	Type: Unsigned16 Classification: Constant Plage valide: > 0	N/A
Réservée pour un usage futur	0	-	-	-
Concentrator ContentRevision	1	Suit les changements apportés au contenu souscrit. Assure que l'objet éditeur Concentrator et l'objet souscrivant Dispersion sont en harmonie	Type: Unsigned8 Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: 0	Mis à jour lorsque le complément des données à éditer change. En cas de ContentRevision non concordant (Tableau 347), la publication ne doit pas être traitée
CommunicationEndpoint	2	Point d'extrémité de l'objet concentrateur qui édite des données vers l'objet dispersion en question	Type: Structure de point d'extrémité d'association de communication Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: L'élément valide de point d'extrémité de connexion configuré indique non configuré (c'est-à-dire que le point d'extrémité n'est pas valide) Plage valide: Voir la définition de la structure	Ecrire dans cet attribut en dernier lors de la configuration de cet objet
MaximumItemsSubscribing	3	Nombre maximal d'éléments qui peuvent être souscrits	Type: Unsigned8 Classification: Constant Accessibilité: Lecture seule Valeur par défaut: Initiative locale Plage valide: >0	Nombre maximal d'éléments dans la publication correspondante
NumItemsSubscribing	4	Nombre d'éléments auxquels s'abonner	Type: Unsigned8 Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: 0	Nombre réel d'éléments dans la publication correspondante Une valeur de zéro indique que l'objet n'est pas configuré pour s'abonner

Tableau 258 *(2 de 2)*

Nom du type d'objet normalisé: Objet Dispersion				
Identificateur du type d'objet normalisé: 5				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Array of ObjectAttributeIndexAndSize	5	Matrice de données pour identifier chaque élément de données publié	Type: Array of ObjectAttributeIndexAndSize Classification : Static Accessibilité: Lecture/écriture Valeur par défaut: La taille de l'élément est 0	ID d'objet, ID d'attribut, indice d'attribut, et taille de données pour la destination au sein de l'application pour les informations éditées NOTE Pour sauter des données, l'objet de destination et l'attribut peuvent localement représenter un objet Null et un attribut Null.
Réservé pour usage futur par la présente norme	6..63	-	-	-

Les attributs Revision, NumItemsSubscribing et d'ObjAttrIdx peuvent être mis en œuvre dans un appareil afin qu'ils puissent être écrits d'une manière atomique dans une transaction de réseau unique par l'intermédiaire de la concaténation des APDU.

12.15.2.6.3 Méthodes d'objets normalisés

Un objet Dispersion présente les méthodes définies dans le Tableau 259.

Tableau 259 – Méthodes d'objet Dispersion

Nom du type d'objet normalisé: Objet Dispersion		
Identificateur du type d'objet normalisé: 5		
Nom de méthode	**ID de méthode**	**Description de la méthode**
Null	0	Réservé par la présente norme pour usage futur
Réservé pour usage futur par la présente norme	0..127	Ces identificateurs de méthode sont réservés pour usage futur par la présente norme
Implementation-specific use	128..255	Ces identificateurs de méthode sont disponibles pour une utilisation spécifique à une mise en œuvre

12.15.2.7 Objet Tunnel

12.15.2.7.1 Généralités

L'objet tunnel (TUN) est utilisé pour prendre en charge le transport à haut rendement énergétique de messages encapsulés sur le réseau pour un seul protocole non natif. Le service de tunnel et une variation du service de publication sont définis pour cette encapsulation. Des structures de prise en charge sont fournies pour la déconstruction, le mapping et la reconstruction des paquets de protocoles non natifs afin de réduire les transactions et la taille de paquet.

NOTE L'utilisation de l'objet tunnel pour créer des convertisseurs de protocole est prévue pour être définie par l'organisation qui a défini le protocole natif utilisé dans le tunnel.

12.15.2.7.2 Attributs d'objet

Un objet tunnel a les attributs définis dans le Tableau 260.

Tableau 260 – Attributs d'objet Tunnel *(1 de 3)*

Nom du type d'objet normalisé: Objet Tunnel				
Identificateur du type d'objet normalisé: 6				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
ObjectIdentifier	Identificateur-clé d'objet	Identificateur unique pour l'objet	Type: Unsigned16 Classification: Constant Plage valide: > 0	N/A
Réservée pour un usage futur	0	-	-	-
Protocole	1	Type de protocole pris en charge par cet objet	Type: Unsigned8 Classification: Constant Accessibilité: Lecture seule Valeur par défaut: Initiative locale (spécifique au protocole) Plage valide: Voir Annexe M	Etablit le protocole spécifique qui est encapsulé dans des messages de tunnel. Seuls les tunnels de protocoles concordants échangent des données sensées
Status (configuration status)	2	Statut de configuration de communication de l'objet en question	Type: Unsigned8 Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: 0 Valeurs nommées: 0: pas configuré; 1: configuration valide; 2: configuration invalide	Le statut d'objet n'est pas configuré lorsque le Protocol est mis à None et aucune communication ne se produit. Une fois que l'objet est configuré et un autre protocole est établi, l'objet tente d'appliquer la configuration et change le statut en conséquence
Flow_Type	3	Service de communication utilisé par cet objet	Type: Unsigned8 Classification: Static Accessibilité: Lecture/écriture Valeurs nommées: 0: tunnel à 2 parties; 1: tunnel à 4 parties; 2: éditer; 3: s'abonner	Configure le tunnel pour un type spécifique de communication et de rôle

Tableau 260 *(2 de 3)*

Nom du type d'objet normalisé: Objet Tunnel				
Identificateur du type d'objet normalisé: 6				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Update_Policy	4	Politique de mise à jour de communication périodique pour l'objet en question	Type: Unsigned8 Classification: Static Accessibilité: Lecture/écriture Valeurs nommées: 0: périodique; 1: changement d'état	Etablit la politique de publication périodique pour un éditeur et un abonné liés. Une mise à jour périodique édite à chaque occasion. Le changement de l'état édite sur des données fraîches ou au moins aussi souvent que le spécifie Stale_Limit
Period (période de publication de données)	5	Période de mise à jour de communication périodique pour l'objet en question	Type: Integer16 Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: 0 Valeurs nommées: 0: non configuré	Etablit le temps de publication périodique pour un éditeur et un abonné liés. La publication isochrone est activée par une règle implicite. La publication ne commence pas tant qu'une période n'est pas établie. Voir 12.12.5
Phase (phase de publication idéale)	6	Phase de communication périodique au sein de la période pour l'objet en question	Type: Unsigned8 Classification: Static Accessibilité: Lecture/écriture Plage valide: 0..99	Etablit le temps de publication demandé au sein de la période pour un éditeur et un abonné liés. La phase réelle peut différer par des exigences de contrat. Il convient que les unités soient indiquées sous forme de pourcentage (%)
Stale_Limit (limite de données périmées)	7	Limite de données périmées de communication périodique pour cet objet	Type: Unsigned8 Classification: Static Accessibilité: Lecture/écriture	Définit le temps d'arrivée maximal attendu de l'abonné comme un multiple de la période. Définit la fréquence minimale de publication pour le rapport relatif au changement d'état comme un multiple de la période
Max_Peer_Tunnels	8	Nombre maximum de tunnels de correspondants avec lesquels cet objet peut communiquer	Type: Unsigned8 Classification: Constant Accessibilité: Lecture seule	N/A

Tableau 260 *(3 de 3)*

Nom du type d'objet normalisé: Objet Tunnel				
Identificateur du type d'objet normalisé: 6				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Num_Peer_Tunnels	9	Nombre réel de tunnels de correspondants avec lesquels cet objet peut communiquer	Type: Unsigned8 Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: 0	Incrémenté/décrémenté à mesure que des éléments de matrice de points d'extrémité de tunnel sont ajoutés et supprimés
Matrice de points d'extrémité de tunnel	10	Matrice de points d'extrémité d'association de protocoles	Type: Matrice de points d'extrémité de tunnel Classification: Static Accessibilité: Lecture/écriture Plage valide: Informations d'adresse pointant vers un ou plusieurs objets tunnel	Relie des objets tunnel distants pour la communication avec cet objet tunnel
Foreign_Source_Address	11	L'adresse source étrangère mise en correspondance avec la communication de cet objet	Type: IPv6Address Classification: Static Accessibilité: Lecture/écriture	Détient des informations d'adressage "static" devant être livrées à l'initiateur ou au correspondant à la réception du message
Foreign_Destination_Address	12	L'adresse de destination étrangère mise en correspondance avec la communication de cet objet	Type: IPv6Address Classification: Static Accessibilité: Lecture/écriture	Détient des informations d'adressage "static" devant être livrées à l'initiateur ou au correspondant à la réception du message
Connection_Info[ ]	13	Informations de connexion étrangères mises en correspondance avec la communication de cet objet	Type: OctetString Classification: Static Accessibilité: Lecture/écriture	Détient des informations "static" devant être livrées à l'initiateur ou au correspondant à la réception du message
Transaction_Info[ ]	14	Informations de transaction étrangères mises en correspondance avec la communication de cet objet	Type: OctetString Classification: Dynamic Accessibilité: Lecture/écriture	Détient des informations spécifiques à la transaction devant être livrées à un initiateur à l'achèvement d'une transaction
Réservé pour usage futur par la présente norme	15..63	-	-	

12.15.2.7.3 Méthodes d'objets normalisés

Un objet tunnel a les méthodes définies dans le Tableau 261.

Tableau 261 – Méthodes d'objet Tunnel

Nom du type d'objet normalisé: Objet Tunnel		
Identificateur du type d'objet normalisé: 6		
Nom de méthode	**ID de méthode**	**Description de la méthode**
Null	0	Réservé par la présente norme pour usage futur
Réservé pour usage futur par la présente norme	0..127	Ces identificateurs de méthode sont réservés pour usage futur par la présente norme
Implementation-specific use	128..255	Ces identificateurs de méthode sont disponibles pour une utilisation spécifique à une mise en œuvre

12.15.2.8 Objet d'interface

12.15.2.8.1 Généralités

L'objet d'interface fournit un point d'extrémité de messagerie générique pour l'interfaçage à un réseau. Cet objet peut être utilisé comme l'objet source ou de destination dans des interactions de messagerie natives exigées pour la prise en charge de la conversion de protocoles de passerelle et diverses applications de messagerie natives.

L'objet d'interface peut être indiqué dans des services client/serveur communication nécessaires pour les services read, write et execute natifs. L'objet d'interface peut également être référencé comme l'objet client communiquant avec un objet upload/download pour le transfert en masse.

Les communications référençant l'objet d'interface comme client doivent adhérer aux politiques de contrôle d'encombrement client/serveur définies dans la présente norme.

Dans la mesure du possible, les réalisateurs doivent considérer de mettre en tampon les valeurs récupérées de client/serveur plutôt que de créer des communications supplémentaires sur le réseau sans fil afin de récupérer à répétition ces valeurs dans des appareils qui ont besoin de préserver l'énergie. Les objets d'application utilisateur contenus dans les appareils de terrain fournissent des guides, par l'intermédiaire de leurs spécifications de la classification de données d'attribut, concernant quelles informations relatives à un objet il convient de placer en tampon.

NOTE 1 La publication et l'abonnement d'objet natifs sont accomplis en utilisant les objets concentrateur et dispersion.

NOTE 2 La structure réelle du tampon/cache et les exigences locales relatives au traitement des messages vers le cache sont considérées comme relevant d'une initiative locale, ne relevant pas du domaine d'application de la présente norme.

12.15.2.8.2 Attributs d'objet

Un objet d'interface a les attributs définis dans le Tableau 262.

Tableau 262 – Attributs d'objet d'interface

Nom du type d'objet normalisé: Objet d'interface				
Identificateur du type d'objet normalisé: 7				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
ObjectIdentifier	Identificateur-clé d'objet	Identificateur unique pour l'objet	Type: Unsigned16 Classification: Constant Plage valide: > 0	N/A
Réservée pour un usage futur	0	-	-	-
Réservé pour usage futur par la présente norme	1..63	-	-	-

12.15.2.8.3 Méthodes d'objets normalisés

Un objet d'interface a les méthodes définies dans le Tableau 263.

Tableau 263 – Méthodes d'objet d'interface

Nom du type d'objet normalisé: Objet d'interface		
Identificateur du type d'objet normalisé: 7		
Nom de méthode	**ID de méthode**	**Description de la méthode**
Null	0	Réservé par la présente norme pour usage futur
Réservé pour usage futur par la présente norme	0..127	Ces identificateurs de méthode sont réservés pour usage futur par la présente norme
Implementation-specific use	128..255	Ces identificateurs de méthode sont disponibles pour une utilisation spécifique à une mise en œuvre

12.16 Types de données

12.16.1 Types de données de base

Les types de données de base pris en charge pour les attributs sont:

- les valeurs binaires;
- les entiers signés de 8 bits, de 16 bits et de 32 bits;
- les entiers non signés (unsigned) de 8 bits, de 16 bits, de 32 bits, de 64 bits et de 128 bits;
- les valeurs virgule flottante (floating point) de 32 bits et de 64 bits selon l'ISO/IEC/IEEE 60559 (IEEE 754);
- les chaînes représentant un texte visible, un bloc d'octets, ou une séquence de valeurs de bits (BitString);
- le temps: TAINetworkTime, TAITimeDifference, TAITimeRounded.

12.16.2 Types de données atomiques dérivées

Les types de données atomiques dérivées pris en charge pour les attributs sont:

- adresses:
  - adresse IPv6Address (mappée à un entier non signé de 128 bits);

  - adresse EUI64Address (mappée à un entier non signé de 64 bits);
  - adresse DL16Address (mappée à un entier non signé de 16 bits);
- identificateurs spécifiques à une couche (généralement mappés à des entiers non signés de 8 bits ou 16 bits);
- indices MIB (généralement mappés à des entiers non signés de 8 bits ou 16 bits).

12.16.3 Structures de données normalisées indépendantes vis-à-vis de toute industrie

12.16.3.1 Généralités

Les structures de données normalisées utilisées doivent être les structures de données acheminées par le protocole défini par la présente norme. Les structures de données normalisées indépendantes vis-à-vis de toute industrie sont résumées en Annexe L.

NOTE Les définitions de structures de données spécifiques à un fournisseur ne sont pas prises en charge.

12.16.3.2 Objet, attribut, indice et taille

Les éléments de ObjectAttributeIndexAndSize sont montrés dans le Tableau 264.

Tableau 264 – Type de données: ObjectAttributeIndexAndSize

Nom du type de données normalisé: ObjectAttributeIndexAndSize		
Code du type de données normalisé: 469		
Nom de l'élément	**Identificateur de l'élément**	**Type scalaire de l'élément**
ObjectID	1	Type: Unsigned16 Classification: Static Accessibilité: Varie selon l'utilisation
AttributeID	2	Type: Unsigned16 Classification: Static Accessibilité: Varie selon l'utilisation
AttributeIndex	3	Type: Unsigned16 Classification: Static Accessibilité: Varie selon l'utilisation
Taille	4	Type: Unsigned16 Classification: Static Accessibilité: Varie selon l'utilisation NOTE Dans la pratique, cette taille maximale dépend des capacités de l'appareil.

12.16.3.3 Communication association endpoint (point d'extrémité d'association de communication)

La structure de données montrée dans le Tableau 265 est utilisée pour des points d'extrémité de communication tant pour les entrées que pour les sorties.

Tableau 265 – Type de données: Communication association endpoint (point d'extrémité d'association de communication) *(1 de 2)*

Nom du type de données normalisé: Point d'extrémité d'association de communication		
Code du type de données normalisé: 468		
Nom de l'élément	**Identificateur de l'élément**	**Type scalaire de l'élément**
Network address of remote endpoint	1	Type: IPv6Address Il s'agit d'une construction logique configurée pour l'appareil par le gestionnaire de système NOTE Le gestionnaire de système s'assure qu'une telle configuration prend en charge le remplacement d'appareil et les scénarios mobiles d'appareil. Classification: Static Accessibilité: Lecture/écriture
T-port at remote endpoint	2	Type: Unsigned16 Classification: Static Accessibilité: Lecture/écriture
Object ID at remote endpoint	3	Type: Unsigned16 Classification: Static Accessibilité: Lecture/écriture
Stale data limit	4	Type: Unsigned8 Classification: Static Accessibilité: Lecture/écriture NOTE 1 Cet attribut a principalement un intérêt pour un abonné. NOTE 2 Le compte de valeurs d'entrée périmées consécutives qu'un abonné n'arrive pas à recevoir avant que l'abonné ne considère la valeur reçue précédemment comme étant non valide (Tableau 299). La péremption est indiquée par un numéro de séquence de fraîcheur inchangé (Tableau 347).
Data publication period	5	Type: Integer16 Classification: Static Accessibilité: Lecture/écriture NOTE Pour les unités de temps, voir 12.12.5
Ideal publication phase	6	Type: Unsigned8 Classification: Static Accessibilité: Lecture/écriture Plage valide: 0..99 (comme un pourcentage %) NOTE Cet attribut a principalement un intérêt pour un éditeur.
PublishAutoRetransmit	7	Type: Unsigned1 Classification: Static Accessibilité: Lecture/écriture Valeurs nommées[a] 0: émettre seulement si le contenu d'application a changé depuis la dernière publication; 1: émettre à chaque occasion périodique (indépendamment de la question de savoir si, oui ou non, le contenu d'application a changé depuis la dernière émission)

Tableau 265 *(2 de 2)*

Nom du type de données normalisé: Point d'extrémité d'association de communication		
Code du type de données normalisé: 468		
Nom de l'élément	**Identificateur de l'élément**	**Type scalaire de l'élément**
Configuration status	8	Unsigned8 Classification: Static Accessibilité: Read access Valeurs nommées: 0: pas configuré (point d'extrémité de connexion non valide); 1: configuré (point d'extrémité de connexion valide) NOTE Le propriétaire des données met cet élément à une valeur de 0 pour indiquer que le point d'extrémité n'est pas configuré, et à 1 pour indiquer que le point d'extrémité est configuré. Un point d'extrémité est considéré comme étant non configuré si la valeur de l'ID d'objet au point d'extrémité distant est 0.
a Le codage de cet attribut est l'inverse de l'attribut associé 12 du Tableau 27.		

12.16.3.4 Données de contrat de communication

La structure de données montrée dans le Tableau 266 est utilisée pour les données dynamiques importantes pour le processus d'application local qui est associé à un contrat de communication particulier.

NOTE 1 Il relève d'une initiative locale de s'assurer que les applications sont bien dressées en termes des contrats de communication qu'elles utilisent. L'AL ne normalise pas les mesures de contrôle relatives à la conformité aux contrats demandés.

NOTE 2 Comme partie intégrante de la négociation de contrat, de l'information suffisante est fournie à l'appareil demandeur de contrat afin de lui permettre de déterminer l'APDU de taille maximale que le contrat prend en charge. Par exemple, si la négociation de contrat détermine la taille maximale d'unité de données de service de réseau (NSDU), le type de sécurité en vigueur pour le contrat est localement déterminable pour le contrat. Si le type de sécurité en cours d'utilisation est connu, la taille d'en-tête de transport est acquise localement et soustraite de la taille de NPDU maximale, donnant ainsi la valeur pour la taille d'APDU maximale utilisable pour des communications utilisant ce contrat de communication particulier.

Tableau 266 – Type de données: Données de contrat de communication

Nom du type de données normalisé: Données de contrat de communication		
Code du type de données normalisé: 470		
Nom de l'élément	**Identificateur de l'élément**	**Type de l'élément**
ContractID	1	Type: Unsigned16 Classification: Static Accessibilité: Lecture seule Plage valide: L'ensemble de valeurs valides est défini par la gestion de système
Contract_Status	2	Type: Unsigned8: Classification: Static Accessibilité: Lecture seule Valeur par défaut = 0 Valeurs nommées: 0: endpoint_not_configured, 1: awaiting_contract_establishment, 2: contract_active_as_requested, 3: contract_active_negotiated_down, 4: awaiting_contract_termination, 5: contract_establishment_failed, 6: contract_inactive
Actual_Phase	3	Type:Unsigned8 Classification: Dynamic Accessibilité: Lecture seule Valeur par défaut: 0 (indiquant non assignée) Plage valide: 0..99 (en unités en pourcentage %)
Des informations supplémentaires sur le contrat réel, telles que les paramètres négociés à la baisse, peuvent être disponibles auprès du DMAP et peuvent ne pas être maintenues par l'UAP.		

12.16.3.5 Point d'extrémité de communication d'alertes

La structure de données montrée dans le Tableau 267 est utilisée pour des points d'extrémité de communication pour les rapports d'alertes.

Tableau 267 – Type de données: Point d'extrémité de communication d'alertes

Nom du type de données normalisé: Point d'extrémité de communication d'alertes		
Code du type de données normalisé: 471		
Nom de l'élément	**Identificateur de l'élément**	**Type scalaire de l'élément**
Network address of remote endpoint	1	Type: IPv6Address Il s'agit d'une construction logique configurée pour l'appareil par le gestionnaire de système NOTE Le gestionnaire de système s'assure qu'une telle configuration prend en charge le remplacement d'appareil et les scénarios mobiles d'appareil. Classification: Static Accessibilité: Lecture/écriture
T-port at remote endpoint	2	Type: Unsigned16 Classification: Static Accessibilité: Lecture/écriture
Object ID at remote endpoint	3	Type: Unsigned16 Classification: Static Accessibilité: Lecture/écriture

12.16.3.6 Point d'extrémité de tunnel

La structure de données montrée dans le Tableau 268 est utilisée dans des objets tunnel pour identifier des points d'extrémité de tunnel distants en vue de l'échange de charges utiles encapsulées.

Tableau 268 – Type de données: Point d'extrémité de tunnel

Nom du type de données normalisé: Point d'extrémité de tunnel		
Code du type de données normalisé: 475		
Nom de l'élément	**Identificateur de l'élément**	**Type scalaire de l'élément**
Network_Address (adresse réseau du point d'extrémité distant)	1	Type: IPv6Address Il s'agit d'une construction logique configurée pour l'appareil par le gestionnaire de système NOTE Le gestionnaire de système s'assure qu'une telle configuration prend en charge le remplacement d'appareil et les scénarios mobiles d'appareil. Classification: Static Accessibilité: Lecture/écriture
Transport_Port (port T au point d'extrémité distant)	2	Type: Unsigned16 Classification: Static Accessibilité: Lecture/écriture
OID (ID d'objet au point d'extrémité distant)	3	Type: Unsigned16 Classification: Static Accessibilité: Lecture/écriture

12.16.3.7 Descripteur de rapports d'alertes

Les éléments du descripteur de rapports d'alertes sont montrés dans le Tableau 269.

Tableau 269 – Type de données: Alert report descriptor (descripteur de rapports d'alertes)

Nom du type de données normalisé: Descripteur de rapports d'alertes		
Code du type de données normalisé: 499		
Nom de l'élément	**Identificateur de l'élément**	**Type de l'élément**
Alert report disabled	1	Type: Boolean8 Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: Initiative locale
Alert report priority	2	Type: Unsigned8 Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: 0 Plage valide: 0..15, comme spécifié en 12.17.5.2.2.22

12.16.3.8 Descripteur d'alarme analogique

Le descripteur d'alarme analogique est utilisé pour définir la production de rapports d'alarme pour une valeur analogique avec une seule condition de référence. Ses éléments sont montrés dans le Tableau 270.

Tableau 270 – Type de données: Descripteur de rapports d'alarmes de contrôle de processus pour analogique avec une seule condition de référence

Nom du type de données normalisé: Descripteur de rapports d'alarmes de contrôle de processus pour analogique avec une seule condition de référence		
Code du type de données normalisé: 498		
Nom de l'élément	**Identificateur de l'élément**	**Type scalaire de l'élément**
Alert report disabled	1	Type: Boolean8 Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: true
Alert report priority	2	Type: Unsigned8 Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: 0 Plage valide: 0..15
Alarm limit	3	Type: Float32 Classification: Static Accessibilité: Lecture/écriture

12.16.3.9 Descripteur d'alarme binaire

Le descripteur d'alarme binaire est la même structure que le type de données pour le descripteur de rapports d'alertes, si bien qu'aucune description supplémentaire de type de données n'est exigée.

12.16.3.10 ObjectIDandType

Les éléments de ObjectIDandType sont montrés dans le Tableau 271.

Tableau 271 – Type de données: ObjectIDandType

Nom du type de données normalisé: ObjectIDandType		
Code du type de données normalisé: 472		
Nom de l'élément	**Identificateur de l'élément**	**Type scalaire de l'élément**
ObjectID	1	Unsigned16
ObjectType	2	Unsigned8
ObjectSubType	3	Unsigned8
VendorSubType	4	Unsigned8

12.16.3.11 Correspondant non programmé

Les éléments de UnscheduledCorrespondent sont montrés dans le Tableau 272.

Tableau 272 – Type de données: Correspondant non programmé

Nom du type de données normalisé: Correspondant non programmé		
Code du type de données normalisé: 473		
Nom de l'élément	**Identificateur de l'élément**	**Type scalaire de l'élément**
Adresse	1	IPv6Address
Port T	2	Unsigned16

12.17 Services d'application fournis par la sous-couche d'application

12.17.1 Généralités

Toutes les interfaces entre la DLE et les entités de couche (ou de sous-couche) ou entités de gestion adjacentes sont des interfaces internes au sein de l'appareil, et sont donc inobservables. Par conséquent, elles ne sont pas soumises à une normalisation.

Des services d'application sont fournis par l'ASL (au niveau du SAP d'ASLDE-n) pour la communication avec des objets natifs, qui sont soit des objets d'UAP, soit des objets de MP. Ce sont les seuls services qu'il convient d'utiliser pour le comportement conforme à la présente norme. Tous les appareils n'ont pas nécessairement besoin d'utiliser l'ensemble des services définis dans le présent document, et tous les objets ne prendront pas nécessairement en charge tous les services dans le présent document. Cependant, si ces services sont utilisés pour la communication entre ou parmi des objets natifs, il convient de les employer tels que définis dans la présente norme.

Il convient que les processus d'application utilisant des services d'ASL soient conçus de façon à tolérer la réception de doublons d'indications et de confirmations de services de l'ASL. Par exemple, si un acquittement de couche inférieure est perdu lorsqu'une réponse à une demande de lecture est envoyée, la couche inférieure peut répéter la tentative et, en conséquence, le client d'application peut recevoir un doublon de réponse à la demande de lecture.

Il est laissé à l'appareil la détermination de la meilleure manière de traiter l'encombrement/la contre-pression si cela est indiqué localement par la suite locale inférieure de protocole. Ce traitement peut, par exemple, limiter l'émission des messages à partir de l'appareil pendant une certaine période de temps, ou pour un certain ensemble de priorités de communication,

ou les deux. L'encombrement peut se produire, par exemple, dans des situations de charge de communication de réseau, et le traitement par l'appareil est destiné à limiter l'encombrement supplémentaire.

NOTE 1 La planification de capacité est une question de système et ne relève pas de la portée d'AL.

Le Tableau 273 résume les services fournis.

NOTE 2 Les services locaux qui ne se traduisent pas en communication réseau ne sont pas inclus dans le Tableau 273, car ils relèvent d'une initiative locale et, donc, sont dépendants d'une mise en œuvre.

NOTE 3 La confirmation locale de service d'ASL renvoyée à l'AP relève d'une initiative locale, et par conséquent elle n'est pas définie par la présente norme.

Tableau 273 – Services d'AL

Service fourni par l'ASL	Primitives applicables	Description	Mode d'utilisation
Services d'accès d'objet			
Read	Request Indication Response Confirmation	Lire une valeur d'attribut dans un objet	Client/server
Write	Request Indication Response Confirmation	Ecrire une valeur dans un objet	Client/server
Execute	Request Indication Response Confirmation	Exécuter une méthode sur un objet	Client/server
Services de publication			
Publish	Request Indication	Editer une ou plusieurs valeurs provenant d'un seul objet source	Publish/subscribe (éditer/s'abonner) NOTE Le contenu natif ainsi que le contenu non natif sont pris en charge
Services relatifs aux rapports d'alertes			
AlertReport	Request Indication	Rapporter une alerte	Source/puits (monodiffusion seulement). La source de ce service doit seulement être l'ARMO. Le puits de ce service doit seulement être l'objet récepteur d'alarme
AlertAcknowledge	Request Indication Response Confirmation	Acquitter une réception d'alerte individuelle	Client/serveur La source de ce service peut seulement être l'objet récepteur d'alarme
Prise en charge explicite pour la tunnellisation			
Tunnel	Request Indication Response Confirmation	Charge utile de tunnel sans analyse d'ASL (pour la compatibilité de protocoles non natifs)	Charge utile de tunnel sans analyse d'ASL (pour la compatibilité de protocoles non natifs). Ce service doit être utilisé seulement si les objets source et destination sont des objets tunnel tous les deux

NOTE 4 Si une demande de service local est mal formée, le fait de rapporter l'erreur à l'UAP demandeur relève d'une initiative locale et n'est donc pas traité dans la présente norme. Si souhaité, une mise en œuvre peut conserver les statistiques concernant les demandes de service localement reçues qui sont mal formées.

12.17.2 Modèle de communication d'application P/S (éditer/s'abonner)

La publication est un processus de communication qui est initié par un objet dans l'UAL éditrice et reçu par un objet dans l'UAL abonnée. La publication utilise des services d'ASL spécifiques à la publication de prise en charge. La publication se produit d'un objet d'éditeur vers un objet abonné. Tout objet peut agir comme éditeur ou comme abonné. Pour optimiser l'utilisation de largeur de bande de communication, des objets spéciaux (un objet concentrateur pour l'édition et un objet dispersion pour l'abonnement) sont définis pour la publication depuis/vers un ensemble d'objets dans un seul UAP en utilisant une seule invocation du service publish.

La raison sémantique de l'édition est purement un souci d'application. Ce modèle prend en charge la communication pour:

- les communications tamponnées périodiques déclenchées par un programme;
- les communications tamponnées déclenchées par une application; et
- les communications tamponnées déclenchées par un changement d'état.

Toutes les communications éditées ci-dessus utilisent le paradigme de flux de communication P/S. Pour des communications périodiques programmées, les applications d'abonné peuvent prendre en charge des méthodes de réponse de temporisation pour traiter une perte de publications individuelles et/ou une perte du point d'extrémité d'éditeur. Par exemple, une application d'abonnement peut utiliser le précédent contenu d'une valeur de publication, mais peut dégrader la qualité correspondante de la valeur. La perte de messages individuels peut avoir une plus courte temporisation que la temporisation utilisée par l'abonné pour déterminer la perte de l'éditeur.

La coordination d'une publication avec le programme de réseau est accomplie par l'intermédiaire de la configuration appropriée de point d'extrémité, qui pilote les demandes de contrat de communication. Une demande de contrat de communication est utilisée pour demander que le gestionnaire de système alloue la largeur de bande programmée pour la communication d'édition.

NOTE 1 La détermination de la politique réelle de temporisation lorsqu'une publication attendue n'est pas reçue est une question spécifique à un processus d'application qui n'est pas spécifiée par la présente norme.

NOTE 2 Les messages édités avec le contenu natif contiennent toujours un compteur de séquence et la/les valeur(s) de données courante(s). S'il n'y a aucun changement de valeur, le compteur de séquence indique que l'application d'édition fonctionne encore et n'a aucune nouvelle donnée à rapporter (cela s'appelle également un heartbeat, c'est-à-dire un signal de présence). Quelques appareils récepteurs maintiennent ce compteur de séquence pour déterminer si la valeur a changé afin de limiter le retraitement, alors que d'autres appareils récepteurs choisissent d'ignorer le compteur de séquence.

NOTE 3 Pour des communications périodiques programmées, les processus d'application utilisent souvent le temps de réseau commun pour synchroniser leurs activités à travers le réseau. Cette synchronisation est localement applicable par des éditeurs et des abonnés pour synchroniser leurs activités au programme de publication.

NOTE 4 Les données et la mesure continues dans la présente norme utilisent un modèle de communication P/S éprouvé de champ de système de commande et effectuent un effet de levier de la largeur de bande programmée pour une synchronisation de communication plus précise. S'il y a une exigence de communication plus personnalisée, elle est considérée comme étant une situation personnalisée ne relevant pas du domaine d'application de la présente norme ou possiblement comme étant une situation pour une considération dans le futur.

12.17.3 Communication tamponnée périodique programmée

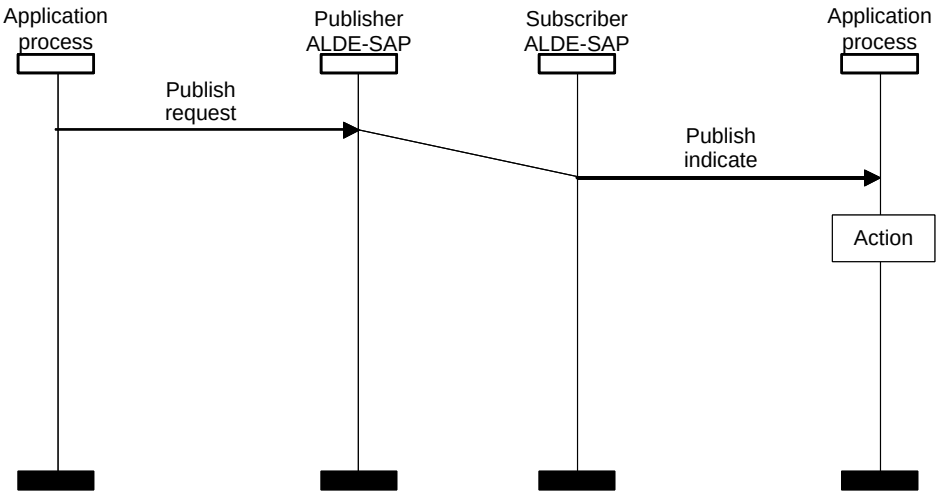
12.17.3.1 Généralités

Le service publish est utilisé pour la communication tamponnée unidirectionnelle vers tout au plus un abonné.

Il convient de configurer l'édition selon les capacités du système. Si un abonné n'est pas présent, cela représente généralement une situation provisoire ou d'erreur et n'est pas censé perdurer. La perte potentielle d'énergie en raison d'appareils configurés incorrectement ou défectueux peut être traitée par une reconfiguration ou par un remplacement d'appareil. Il existe de rares situations anormales pour lesquelles l'optimisation de la conception n'est pas exigée; par conséquent, la complexité exigée pour la publication seulement en présence d'un abonné réel l'emporte sur les éventuelles économies de communications.

Il n'est appliqué aucun acquittement ou aucune répétition de tentative aux interactions P/S. Une ASL éditrice reçoit une demande de service non confirmé et construit une APDU appropriée, qui est alors passée aux couches communication inférieures en vue du transfert de communication. Si une ASL éditrice reçoit une demande pour un service avant que la demande antérieure n'ait été acheminée, il convient que la nouvelle demande écrase la demande antérieure et il convient de ne pas émettre la demande antérieure. Si une ASL abonnée reçoit une nouvelle demande avant que la demande précédente n'ait été livrée à l'objet de destination, la nouvelle demande écrase la demande précédente, et la demande précédente est perdue.

Les services définis prennent en charge la publication d'un attribut arbitraire (une variable de processus, par exemple) issu d'un seul appareil, ou la publication d'un jeu d'attributs issus d'un appareil sans fil plus complexe (à capacité de plusieurs variables, par exemple). Les communications P/S ont lieu sur une relation de communication configurée, telle que montrée à la Figure 125.



Légende

Anglais	Français
Application process	Processus d'application
Publish request	Demande d'édition
Publisher ALDE-SAP	ALDE-SAP éditeur
Subscriber ALDE-SAP	ALDE-SAP abonné
Publish indicate	Indication d'édition
Action	Action

Figure 125 – Séquence de publication de primitives de service

L'abonné pour une publication peut être, par exemple, une passerelle.

NOTE 1 Le contenu des publications non natives ne relève pas du domaine d'application de la présente norme.

Les publications natives incluent la valeur d'attribut elle-même, et les informations de statut pour la valeur. Afin de prendre en charge la détection de doublons et la livraison dans le désordre, un simple compteur monotone d'un seul octet est inclus avec chaque valeur éditée.

NOTE 2 D'autres plans pour identifier de façon univoque un message édité, tels que l'utilisation d'un marqueur temporel en lieu et place d'un compteur, avaient été envisagés mais furent éliminés parce qu'ils requièrent plus d'énergie pour communiquer. Le temps de construction de TPDU, combiné à une indication de fraîcheur fournie par une couche inférieure, est souvent disponible localement. Les marqueurs temporels sur des publications de données sont utiles dans les passerelles de mise en tampon style unité terminale distante (RTU), mais pour de telles passerelles, un marqueur temporel de réception est également utilisable dans ce but, en particulier parce que toutes les publications utilisent un programme de saut de voie partagé, laissant une petite variance entre le temps de génération de données et le temps de réception de données.

La priorité de message P/S peut être fixe; dans ce cas, une mise en œuvre locale peut choisir de ne pas fournir la priorité par message. Si une application requiert des publications avec différents niveaux prioritaires, telles qu'une publication de basse priorité pour la surveillance de contrôle et une publication de haute priorité pour des variables de boucles de commande de haut débit (1 Hz et 4 Hz), des relations P/S distinctes peuvent être exigées.

L'éditeur d'AL et l'abonné d'AL ne communiquent pas explicitement pour établir ou rompre leur relation; cependant, l'établissement de relations de communication sécurisées peut forcer l'établissement de relations de transport. Les éditeurs et les abonnés établissent chacun de façon indépendante (asynchrone) leur partie de la relation. C'est-à-dire, soit l'UAP éditeur, soit un nombre quelconque des UAP abonnés, peut agir le premier pour établir sa partie d'une relation P/S. La messagerie de bout en bout ne peut pas débuter avant que l'éditeur et l'(les) abonné(s) n'aient établi leurs côtés respectifs de la relation de communication. Une fois que l'éditeur a créé un côté de la relation de communication et un abonné a créé l'autre côté de la relation de communication, des messages de publication peuvent être envoyés et livrés aux objets d'application correspondants.

Le fait de localiser dynamiquement les éditeurs en utilisant soit un service de découverte d'étiquette, soit un service de consultation d'annuaire centralisé, ne relève pas du domaine d'application de la présente norme. Par conséquent, la P/S est censée être compatible avec un mécanisme de configuration statique. Dans de futures versions, un mécanisme de découverte peut être utilisé. Dans l'une ou l'autre situation, les mêmes informations ont besoin d'être disponibles pour établir la relation P/S.

Des chemins de communication sont formés d'une manière transparente à l'AL

NOTE 3 La formation de chemins d'émission à l'appui de la P/S est importante pour assurer la livraison en temps utile, mais elle est laissée à la responsabilité du gestionnaire de système, qui forme des chemins à utiliser au cours de l'établissement de contrat de communication. Voir 6.3.11 pour plus de détails.

La publication est toujours une demande de service de transfert de données non confirmé. L'abonnement se traduit toujours par la réception d'une indication de service de transfert de données non confirmé.

NOTE 4 Est considéré comme étant un thème de gestion le fait d'assurer que la configuration de sécurité/les changements prennent en charge la P/S sans impact de l'AL. Il est sous-entendu que les considérations de sécurité contraignent souvent les relations permises.

La temporisation d'une publication programmée est coordonnée à travers le réseau et, à ce titre, dépend d'une vue coordonnée du temps à travers le réseau. La largeur de bande allouée pour les publications déclenchées par un programme a besoin d'être réservée pour s'assurer que les abonnés peuvent recevoir ce que leurs éditeurs envoient. Il convient que le programme soit configuré pour assurer le meilleur effort pour satisfaire aux dates limites de livraison, mais finalement, la responsabilité de l'éditeur de créer de nouvelles publications, et celle de l'abonné d'agir à la réception de celles-ci, dépendent de la programmation interne de l'appareil du processus d'application.

Les communications éditées s'appuient sur le support de service de publication fourni par la suite de protocoles de communication qui est sous-jacente à l'ASL. Un aspect important de cette suite inférieure de protocoles de communication est la capacité de fournir la

temporisation de communication spécifique afin de satisfaire aux exigences de programmation.

12.17.3.2 Publish

12.17.3.2.1 Généralités

Le service d'édition pour la présente norme est un service en monodiffusion utilisé pour mettre périodiquement à jour des données, d'une seule source de publication dans un AP (tout au plus) vers une seule destination d'abonné.

Le service d'édition peut également être utilisé d'une façon apériodique pour prendre en charge les modifications tant déclenchées par une application que déclenchées par un changement d'état. Sachant que le contenu d'un tampon est émis selon un programme, la communication éditée native inclut un indicateur de fraîcheur pour permettre à l'abonné de déterminer si, oui ou non, une valeur a changé.

NOTE La fraîcheur ne signifie pas des données inchangées, mais plutôt que la valeur a été nouvellement (fraîchement) acquise depuis qu'elle a été éditée la dernière fois.

Le Tableau 274 définit les primitives de service.

Tableau 274 – Service Publish

Nom du paramètre	Demande	Indication
Argument	M	M
Service contract identifier	M	—
Priority	M	—
Discard eligible	M	—
End-to-end transmission time	—	M
Published data size	M	M
Subscriber T port	M	—
Subscriber TDSAP	—	M
Subscribing object identifier	M	M(=)
Publisher IPv6Address	—	M
Publisher TDSAP	M	—
Publisher T port	—	M(=)
Publishing object identifier	M	M(=)
DataStructureInformation	M	M(=)
NativeIndividualValue	S	S(=)
Freshness sequence number	M	M(=)
Individual analog value and status	S	S(=)
Individual digital value and status	S	S(=)
NativeValueList	S	S(=)
Publishing content version	M	M(=)
List of publish data	M	M(=)
Fresh value sequence number	M	M(=)
Analog value and status	S	S(=)
Digital value and status	S	S(=)
Non-native	S	S(=)
Non-native data	M	M(=)

12.17.3.2.2 Arguments

12.17.3.2.2.1 Service contract identifier

Ce paramètre identifie l'accord de contrat de service de communication qui a été conclu entre l'UAP demandant le service et le DMAP local. La valeur doit être située dans l'ensemble de valeurs valides pour un identificateur de contrat tel que défini par 6.3.11.

12.17.3.2.2.2 Priority

Ce paramètre définit la priorité de message du service qui est exigée de la communication. Les valeurs permises pour ce paramètre de service peuvent être une indication d'un message

de haute priorité ou d'un message de basse priorité. L'émission et la livraison des messages de haute priorité sont plus importantes que l'émission et la livraison des messages de basse priorité.

12.17.3.2.2.3 Discard eligible

Ce paramètre définit les conseils au réseau de communication relatifs à l'impact d'application du fait de rejeter le message d'application en cas d'encombrement de réseau. Les valeurs possibles sont true (le message peut être considéré pour le rejet), ou false (ne pas considérer le message pour le rejet).

NOTE Ces conseils sont fournis pour être utilisés par des routeurs qui sont construits pour utiliser une politique de rejet intelligent de messages plutôt qu'une politique de rejet aléatoire dans des situations d'encombrement de réseau.

12.17.3.2.2.4 End-to-end transmission time

Il s'agit du temps d'émission de la TLE en l'appareil demandeur vers la TLE en l'appareil récepteur. L'intervalle est marqué par deux instants, le premier instant étant la livraison à la TLE dans l'appareil demandeur, et le second instant étant la réception par TLE dans l'appareil de destination.

12.17.3.2.2.5 Published data size

Ce paramètre fournit à l'abonné le nombre d'octets des données à éditer.

12.17.3.2.2.6 Subscriber T-port

Ce paramètre identifie le port T associé de l'UAP d'abonné.

12.17.3.2.2.7 Subscriber TDSAP

Ce paramètre identifie le TDSAP d'abonné associé au port T d'abonné.

12.17.3.2.2.8 Subscribing object identifier

Ce paramètre spécifie la destination de l'identificateur d'objet dans l'application qui est abonnée à cette publication.

12.17.3.2.2.9 Publisher IPv6Address

Ce paramètre identifie l'adresse IPv6Address de l'éditeur.

12.17.3.2.2.10 Publisher TDSAP

Ce paramètre identifie de façon univoque les TDSAP associés des UAP d'éditeur. Le TDSAP établit un mapping 1:1 à un UAP. La valeur doit être un membre du jeu des TDSAP valides, tels que spécifiés par la TL.

12.17.3.2.2.11 Publisher T-port

Ce paramètre identifie le port T associé de l'UAP d'éditeur.

NOTE Une mise en œuvre peut inférer ce paramètre à partir du TDSAP d'éditeur. Il est donc inclus ici pour l'exhaustivité, telle que exigée par la présente norme pour le mapping logique à la définition de demande de service de données de transport.

12.17.3.2.2.12 Publisher object identifier

Ce paramètre identifie l'objet éditeur qui est la source de données éditées.

NOTE S'il y a plus d'une entrée dans la liste de données éditées, la source objet d'édition est une instance du type d'objet concentrateur.

12.17.3.2.2.13 DataStructureInformation

Ce paramètre indique la construction des informations devant être acheminées par l'intermédiaire de la publication. Il peut indiquer l'une des constructions suivantes:

- native individual value (valeur individuelle native);
- native sequence of values (séquence natives de valeurs); ou
- non-native data (données non natives, à savoir, les informations tunnellisées par l'intermédiaire d'un service de publication).

Les détails de ces alternatives se présentent comme suit:

a) La structure de données de chaque valeur native individuelle est comme suit:

1) Freshness value sequence number

Ce paramètre est présent si les informations relatives à la structure de données indiquent que la structure des données est une valeur individuelle native. Ce paramètre indique la fraîcheur des données.

2) Individual analog value and status

Ce paramètre est présent si la valeur native individuelle est une valeur analogique. Il contient la structure de données normalisées de statut de valeur qui indique des informations telles que la qualité de la valeur analogique correspondante et la valeur analogique elle-même.

3) Individual digital value and status

Ce paramètre est présent si la valeur native individuelle est une valeur numérique. Il contient la structure de données normalisées de statut de valeur qui indique des informations telles que la qualité de la valeur numérique correspondante et la valeur numérique elle-même.

b) La structure de données d'une liste de valeurs natives est comme suit:

1) Publishing content version

Ce paramètre est présent si la publication est pour des données natives de la liste, telles qu'envoyées en provenance un objet concentrateur. Ces informations assurent l'interprétation harmonieuse des informations éditées par l'abonné.

2) List of publish data

Ce paramètre représente la liste de données acheminées par l'intermédiaire du service publish.

3) Status and analog value

Il contient la structure de données normalisées de statut de valeur qui indique des informations telles que la qualité de la valeur analogique correspondante et la valeur analogique elle-même.

4) Status and digital value

Il contient la structure de données normalisées de statut de valeur qui indique des informations telles que la qualité de la valeur numérique correspondante et la valeur numérique elle-même.

c) Les données contenues dans le service publish qui sont non natives sont comme suit:

Ce paramètre contient les données non natives à éditer. Des données non natives sont acheminées sous la forme d'une chaîne d'octets.

12.17.4 Interactions client/serveur

12.17.4.1 Généralités

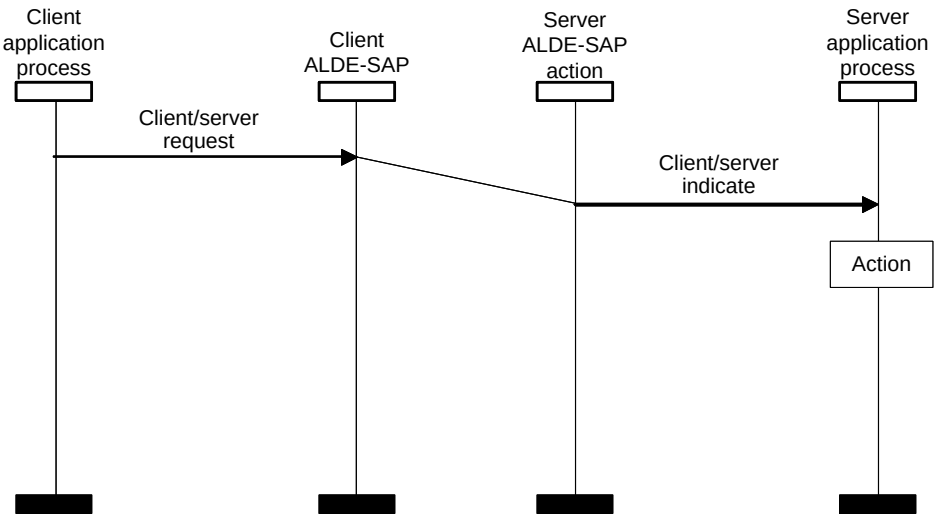
Les interactions client/serveur sont utilisées pour les communications apériodiques une à une. Ces relations utilisent la communication bidirectionnelle placée en file d'attente à la demande. Les services client/serveur définis par la présente norme sont soit un service à deux parties (ayant deux primitives de service, à savoir: .req et .ind), comme à la Figure 126, ou un service à quatre parties (ayant quatre primitives de service, à savoir: .req, .ind, .rsp, et .cnf), comme à la Figure 127, à la Figure 128 et à la Figure 129.

Lorsque l'ASL reçoit une demande de service client/serveur, elle construit une unité de données de protocole d'application (APDU) correspondante et demande un transfert placé en file d'attente auprès de la suite inférieure de protocoles de communication. L'ASL serveur reçoit une indication de réponse de service confirmé, qu'elle délivre à l'UAP et à l'objet de destination.

Pour les services avec des primitives en quatre parties, l'UAP serveur construit une réponse correspondante. Lorsque l'ASL reçoit la réponse de service client/serveur, elle construit une APDU de réponse et la présente à sa suite inférieure de protocoles de communication, qui fournit les services de communications orientés file d'attente pour livrer la réponse.

Dans une interaction client/serveur, chaque point d'extrémité de la communication peut agir comme client et/ou serveur. Une demande client est envoyée vers une seule destination (serveur). Cette demande indique la destination vers laquelle il convient d'envoyer la réponse. Une seule réponse est alors émise en provenance du serveur.

Les interactions telles que montrées à la Figure 127, à la Figure 128 et à la Figure 129 exigent qu'un identificateur de contrat de communication et la suite de protocoles de communication soient correctement configurés pour prendre en charge les exigences de l'application relatives à la messagerie client/serveur. La largeur de bande représentée par l'identificateur de contrat est considérée comme étant une largeur de bande partagée non programmée qui peut ne pas être réservée seulement pour être utilisée par le présent contrat.

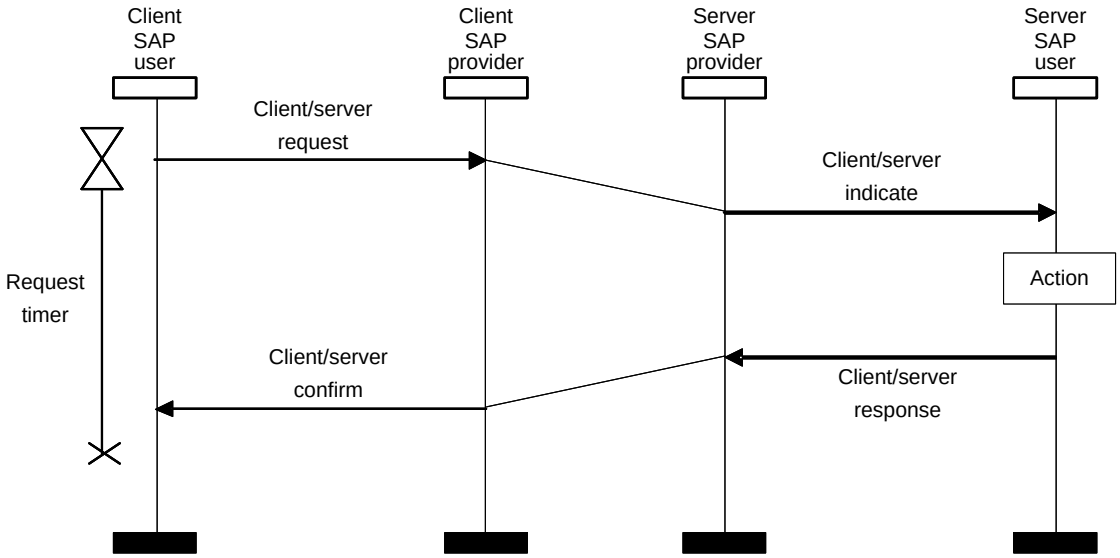


Légende

Anglais	Français
Client application process	Processus d'application client
Client/server request	Demande client/serveur
Client ALDE-SAP	ALDE-SAP client
Server ALDE-SAP action	Action d'ALDE-SAP serveur

Anglais	Français
Client/server indicate	Indication client/serveur
Server application process	Processus d'application serveur
Action	Action

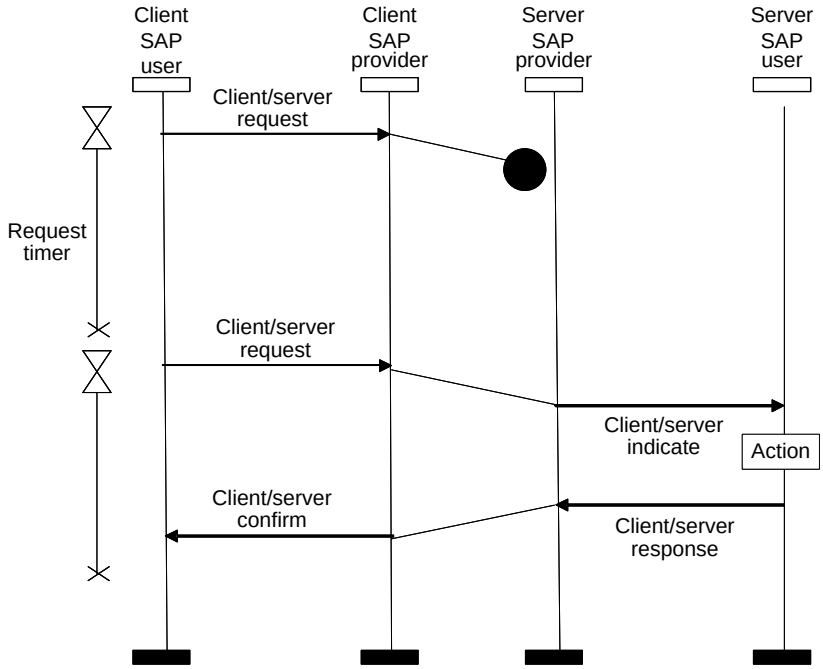
Figure 126 – Modèle de client/serveur, interactions à deux parties



Légende

Anglais	Français
Client SAP user	Utilisateur de SAP client
Client/server request	Demande client/serveur
Client SAP provider	Fournisseur SAP client
Client/server indicate	Indication client/serveur
Server SAP provider	Fournisseur SAP serveur
Server SAP User	Utilisateur de SAP serveur
Action	Action
Request timer	Temporisateur de demande
Client/server confirm	Confirmation client/serveur
Client/server response	Réponse client/serveur

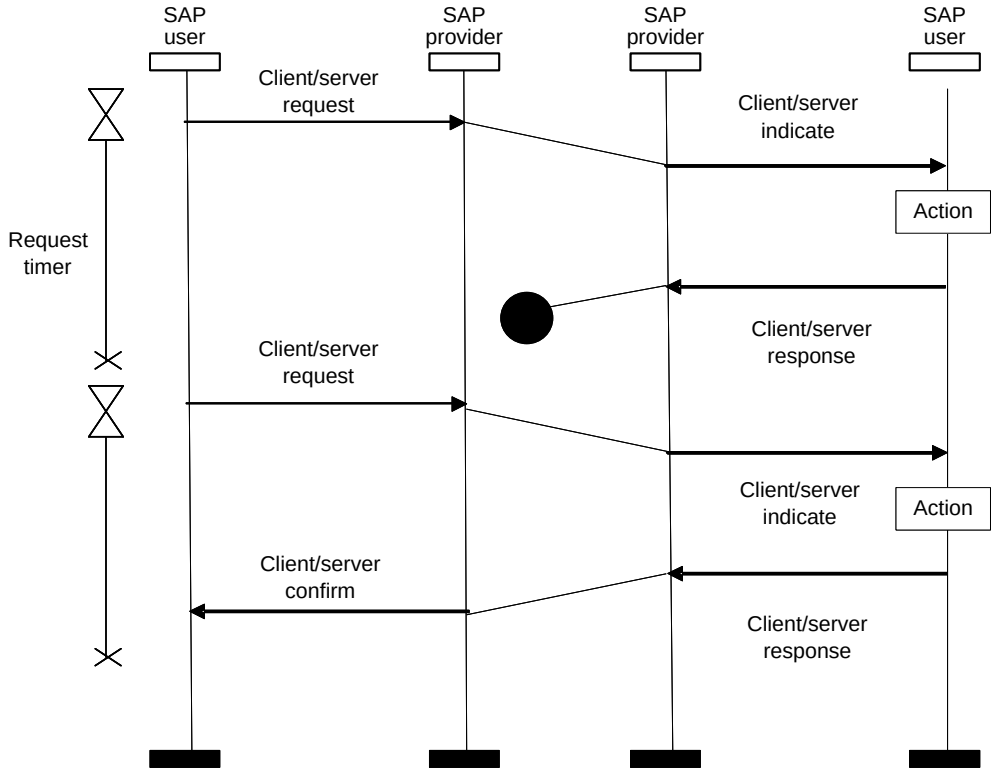
Figure 127 – Modèle de client/serveur interactions à quatre parties: Livraison réussie



Légende

Anglais	Français
Client SAP user	Utilisateur de SAP client
Client SAP provider	Fournisseur SAP client
Server SAP provider	Fournisseur SAP serveur
Server SAP user	Utilisateur de SAP serveur
Client/server request	Demande client/serveur
Request timer	Temporisateur de demande
Client/server response	Réponse client/serveur
Client/server indicate	Indication client/serveur
Client/server confirm	Confirmation client/serveur
Action	Action

Figure 128 – Modèle de client/serveur interactions à quatre parties: Echec de livraison de demande



Légende

Anglais	Français
SAP user	Utilisateur de SAP
SAP provider	Fournisseur SAP
Client/server request	Demande client/serveur
Client/server indicate	Indication client/serveur
Request timer	Temporisateur de demande
Client/server response	Réponse client/serveur
Client/server confirm	Confirmation client/serveur
Action	Action

Figure 129 – Modèle de client/serveur interactions à quatre parties: Echec de livraison de réponse

Lorsqu'une association client/serveur est sécurisée, une session de sécurité est impliquée. Afin d'optimiser l'élimination des connexions que le processus UAL connaît comme n'étant plus exigées, une interface locale pour résilier le contrat peut être utilisée. La résiliation de contrat peut être utilisée pour libérer une session de sécurité (s'il en existe une).

Pour initier la communication, le client demande à envoyer un message à un serveur. Le client spécifie l'identificateur de contrat local qui indique l'adresse IPv6Address du serveur. La communication identifie également l'application et l'objet de source particuliers faisant la demande, l'application et l'objet de destination censés recevoir la demande, ainsi que les informations réelles spécifiques à une instance de service.

Le serveur envoie une réponse au client spécifiant son identificateur de contrat local, qui indique l'adresse IPv6Address du client. La communication spécifie également des informations suffisantes pour livrer la réponse à l'objet d'application approprié et pour collationner la réponse avec la demande originale.

Un contrat de communication doit être établi entre le client et le serveur pour transporter la demande du client, et du serveur vers le client pour transporter la réponse du serveur.

En jouant le rôle d'un client, un client peut envoyer des demandes vers un serveur. En jouant le rôle d'un serveur, un serveur peut envoyer des réponses vers un client. Le client est responsable de la réponse de temporisation de serveur et de l'intégrité transactionnelle.

Un serveur simple pourrait prendre en charge une seule transaction en cours avec un client particulier. Si un retard prolongé se produit pour recevoir une réponse, le client peut, par exemple, temporiser et envoyer de nouveau la demande. Si cela se produit, des doublons de réponses peuvent être reçus. Si un serveur a des ressources pour prendre en charge plusieurs transactions en cours avec un client, les demandes et les réponses peuvent arriver dans le désordre. Pour prendre en charge cette situation, un identificateur de demande est utilisé pour activer/désactiver la collation de demande/réponse.

S'il y a un besoin de plusieurs messages client ou serveur comme partie intégrante d'une séquence de communication, la mise en œuvre peut considérer d'utiliser la concaténation d'ASL. Au-delà de la concaténation, le flux continu de messages est une responsabilité spécifique à un processus d'application ne relevant pas du domaine d'application de la norme.

Les caractéristiques de communications pour des interactions client/serveur telles que la temporisation de réponse relèvent d'une initiative locale, ne relevant pas du domaine d'application de la présente norme. Par exemple, elles peuvent être fixées par la construction d'appareil ou déterminées par un programme d'application au sein d'un appareil, ou être configurées pour l'appareil sur une base par processus d'application ou même sur une base par contrat. Des interactions client/serveur sont habituellement utilisées pour la configuration (telle que la configuration relative à un contrôle de processus ou la configuration d'objet de gestion) et l'échange ad hoc d'informations.

Il convient que les communications client/serveur n'interfèrent pas avec des communications programmées, car il est essentiel que la largeur de bande d'émission soit allouée pour prendre en charge des contrats de communication de messagerie client/serveur. L'intention est d'assurer la capacité de reconfigurer un appareil.

Occasionnellement, seulement un seul échange client/serveur est exigé. Cela peut entraîner un surdébit substantiel pour établir les chemins et la sécurité. A d'autres moments, plusieurs échanges client/serveur se produisent entre les mêmes points d'extrémité.

NOTE Toute altération des chemins de communication (pour compenser l'interférence, par exemple) se produit d'une manière transparente à l'AL.

Le client spécifie la priorité de message désirée pour des demandes de service. Le serveur spécifie la priorité de message désirée pour des réponses de service.

Il convient que les messages de plus haute priorité se déplacent idéalement à l'avant des files d'attente de messages de suite de protocoles d'émission hiérarchisés par ordre de priorité prenant en charge des communications client/serveur. Si possible, il convient que l'allocation de largeur de bande de messages de client/serveur sur le réseau accorde l'accès d'abord aux demandes de message de priorité plus haute. La sécurité est présumée être établie sur une base de contrat et, donc, la sécurité par message n'est pas fournie.

D'autres considérations pour le repli d'émission, telles que basées sur un encombrement de réseau, sont une responsabilité globale d'appareil et ne sont pas une responsabilité spécifique de l'AL.

12.17.4.2 Services client/serveur

12.17.4.2.1 Généralités

Les services suivants sont fournis comme communications client/serveur:

- read;
- write;
- execute; et
- tunnel.

NOTE Tunnel comme primitive à deux parties est également utile pour la communication source/puits.

12.17.4.2.2 Codes de retour d'informations relatives à un service

Les services C/S à quatre parties fournissent un code de retour d'informations relatives à un service pour indiquer le résultat du service du point de vue du serveur. Une plage de codes est réservée pour les additions spécifiques à un fournisseur.

12.17.4.3 Lecture

12.17.4.3.1 Généralités

Le service de lecture est utilisé pour lire un attribut d'un objet dans un processus d'UAL.

Le Tableau 275 définit les primitives du service de lecture.

Tableau 275 – Service de lecture

Nom du paramètre	Demande	Indication	Réponse	Confirmation
Argument	M	M	-	-
Service contract identifier	M	-	-	-
Priority	M	-	-	-
Discard eligible	M	-	-	-
End-to-end transmission time	-	M	-	-
Forward congestion notification	-	M	-	-
Server T-port	M		-	-
Server TDSAP	-	M	-	-
Server object identifier	M	M(=)	-	-
Client IPv6Address	-	M	-	-
Client TDSAP	M	-	-	-
Client T-port	-	M	-	-
Client object identifier	M	M(=)	-	-
Application request ID	M	M(=)	-	-
Data to be read	M	M(=)	-	-
Attribute identifier	M	M(=)	-	-
Attribute index(es)	C	C(=)	-	-
Result	-	-	M	M
Service contract identifier	-	-	M	-
Priority	-	-	M	-
Discard eligible	-	-	M	-
End-to-end transmission time	-	-	-	M
Forward congestion notification	-	-	-	M
Forward congestion notification echo	-	-	M	M(=)
Server IPv6Address	-	-	-	M
Server TDSAP	-	-	M	-
Server T-port	-	-	-	M
Server object identifier	-	-	M	M(=)
Client T-port	-	-	M	-
Client TDSAP	-	-	-	M
Client object identifier	-	-	M	M(=)
Application request ID	-	-	M	M(=)
Value read	-	-	M	M(=)
Service feedback code	-	-	M	M(=)
Value size	-	-	C	C(=)
Data value	-	-	C	C(=)

12.17.4.3.2 Argument

12.17.4.3.2.1 Service contract identifier

Voir 12.17.3.2.2.1.

12.17.4.3.2.2 Priority

Voir 12.17.3.2.2.2.

12.17.4.3.2.3 Discard eligible

Voir 12.17.3.2.2.3.

12.17.4.3.2.4 End-to-end transmission time

Voir 12.17.3.2.2.4.

12.17.4.3.2.5 Forward congestion notification

Ce paramètre indique si la demande a rencontré un encombrement de réseau sur son chemin allant du client vers le serveur.

12.17.4.3.2.6 Server T-port

Ce paramètre identifie l'UAP serveur associé au port T serveur.

12.17.4.3.2.7 Server TDSAP

Ce paramètre identifie le TDSAP serveur associé au port T serveur.

12.17.4.3.2.8 Server object identifier

Ce paramètre identifie un objet serveur à partir duquel il est souhaité lire des données.

12.17.4.3.2.9 Client/source address

Ce paramètre identifie l'adresse IPv6Address pour le client de cette demande.

12.17.4.3.2.10 Client/source TDSAP

Ce paramètre identifie le TDSAP associé de l'UAP client ou serveur. L'UAP contient l'objet qui est à l'origine de la demande.

12.17.4.3.2.11 Client T-port

Ce paramètre identifie l'UAP client associé au port T.

12.17.4.3.2.12 Client object identifier

Ce paramètre identifie l'objet client qui initie la demande de service.

12.17.4.3.2.13 Application request identifier

Identificateur fourni par l'UAP pour représenter de façon univoque cette demande.

12.17.4.3.2.14 Data to be read

Ce paramètre identifie les valeurs de données que le client souhaite lire.

12.17.4.3.2.15 Attribute identifier

Ce paramètre identifie l'attribut de l'objet serveur, dont il est souhaité lire la valeur.

12.17.4.3.2.16 Attribute index/indices

Ce paramètre identifie l'indice ou les indices pour les informations d'intérêt à partir de l'attribut. Il peut y avoir:

- zéro indice, comme dans le cas d'une valeur scalaire;
- un seul indice, par exemple, pour accéder
  - à un élément d'une matrice unidimensionnelle, ou
  - à un membre d'une structure de données normalisée; ou
- deux indices, par exemple, pour accéder
  - à un élément d'une matrice bidimensionnelle ou
  - à un membre d'une structure normalisée contenue dans une matrice unidimensionnelle de structures de données normalisées identique, ou
  - une tranche unidimensionnelle d'une matrice bidimensionnelle, ou

– une tranche unidimensionnelle d'une matrice unidimensionnelle constituée de structures de données normalisées identiques, extrayant sous la forme d'une matrice unidimensionnelle une tranche sectionnelle d'un membre unique à travers ces structures de données identiques.

12.17.4.3.3 Result

12.17.4.3.3.1 Service contract identifier

Voir 12.17.3.2.2.1.

12.17.4.3.3.2 Priority

Voir 12.17.3.2.2.2.

12.17.4.3.3.3 Discard eligible

Voir 12.17.3.2.2.3.

12.17.4.3.3.4 End-to-end transmission time

Voir 12.17.3.2.2.4.

12.17.4.3.3.5 Forward congestion notification

Voir 12.17.4.3.2.5.

12.17.4.3.3.6 Forward congestion notification echo

Ce paramètre indique si la demande de service a rencontré un encombrement de réseau sur son chemin allant du client vers le serveur.

12.17.4.3.3.7 Server IPv6Address

Ce paramètre identifie l'adresse IPv6Address pour le serveur de cette demande.

12.17.4.3.3.8 Server TDSAP

Voir 12.17.4.3.2.7.

12.17.4.3.3.9 Server T-port

Voir 12.17.4.3.2.6.

12.17.4.3.3.10 Server object identifier

Voir 12.17.4.3.2.8.

12.17.4.3.3.11 Client T-port

L'UAP contient l'objet qui est à l'origine de la demande. Voir 12.17.4.3.2.11.

12.17.4.3.3.12 Client TDSAP

L'UAP contient l'objet qui est à l'origine de la demande. Voir 12.17.4.3.2.10.

12.17.4.3.3.13 Client object identifier

Voir 12.17.4.3.2.12.

12.17.4.3.3.14 Application request identifier

Ce paramètre est un identificateur fourni par le client pour représenter de façon univoque cette demande.

12.17.4.3.3.15 Value read

La valeur read indique le résultat de l'opération demandée, et si la lecture est un succès, la taille et la valeur de l'attribut d'objet devant être lu.

12.17.4.3.3.16 Service feedback code

Le service feedback code indique si l'opération demandée a réussi ou pas. Si elle n'a pas réussi, il fournit des informations indiquant pourquoi elle n'a pas réussi.

12.17.4.3.3.17 Value size

Value size indique le nombre d'octets contenus dans la valeur de données. Il est présent si et si seulement si le service feedback code correspondant indique un succès.

12.17.4.3.3.18 Data value

Data value est la valeur de données qui ont été lues dans l'objet serveur, l'attribut et l'indice d'attribut identifiés. Il est présent si et seulement si le service feedback code indique un succès et le paramètre Value size est différent de zéro.

12.17.4.4 Write

12.17.4.4.1 Généralités

Le service d'écriture est utilisé pour écrire une valeur ou un ensemble de valeurs dans un ou plusieurs attributs d'un ou plusieurs objets dans un processus d'application.

Une écriture dans une structure contenant des éléments tant inscriptibles qu'en lecture seule est permise. Dans cette situation, les éléments en lecture seule doivent être inaltérés.

Le Tableau 276 définit les primitives du service d'écriture.

Tableau 276 – Service d'écriture

Nom du paramètre	Demande	Indication	Réponse	Confirmation
Argument	M	M	-	-
Service contract identifier	M	-	-	-
Priority	M	-	-	-
Discard eligible	M	-	-	-
End-to-end transmission time	-	M	-	-
Forward congestion notification	-	M	-	-
Server T-port	M	-	-	-
Server TDSAP	-	M	-	-
Server object identifier	M	M(=)	-	-
Client IPv6Address	-	M	-	-
Client TDSAP	M	-	-	-
Client T-port	-	M	-	-
Client object identifier	M	M(=)	-	-
Application request ID	M	M(=)	-	-
Data to write	M	M(=)	-	-
Attribute identifier	M	M(=)	-	-
Attribute index(es)	M	M(=)	-	-
Value size	M	M(=)	-	-
Data value	M	M(=)	-	-
Result	-	-	M	M
Service contract identifier	-	-	M	-
Priority	-	-	M	-
Discard eligible	-	-	M	-
End-to-end transmission time	-	-	-	M
Forward congestion notification	-	-	-	M
Forward congestion notification echo	-	-	M	M(=)
Server IPv6Address	-	-	-	M
Server TDSAP	-	-	M	-
Server T-port	-	-	-	M(=)
Server object identifier	-	-	M	M(=)
Client T-port	-	-	M	-
Client TDSAP	-	-	-	M
Client object identifier	-	-	M	M(=)
Application request ID	-	-	M	M(=)
Service feedback code	-	-	M	M(=)

12.17.4.4.2 Argument

12.17.4.4.2.1 Service contract identifier

Voir 12.17.3.2.2.1.

12.17.4.4.2.2 Priority

Voir 12.17.3.2.2.2.

12.17.4.4.2.3 Discard eligible

Voir 12.17.3.2.2.3.

12.17.4.4.2.4 End-to-end transmission time

Voir 12.17.3.2.2.4.

12.17.4.4.2.5 Forward congestion notification

Voir 12.17.4.3.3.6.

12.17.4.4.2.6 Server T-port

Voir 12.17.4.3.2.6

12.17.4.4.2.7 Server TDSAP

Voir 12.17.4.3.2.7

12.17.4.4.2.8 Server object identifier

Voir 12.17.4.3.2.8

12.17.4.4.2.9 Client/source address

Ce paramètre identifie l'adresse IPv6Address associée de l'UAP client ou source. L'UAP contient l'objet qui est à l'origine de la demande.

12.17.4.4.2.10 Client/source TDSAP

Ce paramètre identifie le TDSAP associé de l'UAP client ou serveur. L'UAP contient l'objet qui est à l'origine de la demande.

12.17.4.4.2.11 Client T-port

Voir 12.17.4.3.2.11.

12.17.4.4.2.12 Client object identifier

Voir 12.17.4.3.2.12.

12.17.4.4.2.13 Application request identifier

Identificateur fourni par l'UAP pour représenter de façon univoque cette demande.

12.17.4.4.2.14 Data to write

Ce paramètre identifie la valeur d'attribut et de données cibles que le client souhaite écrire.

12.17.4.4.2.15 Attribute identifier

Ce paramètre identifie l'attribut de l'objet serveur, dont il est souhaité lire la valeur.

12.17.4.4.2.16 Attribute index/indices

Ce paramètre identifie l'indice ou les indices pour les informations d'intérêt à partir de l'attribut. Voir 12.17.4.3.2.16.

12.17.4.4.2.17 Value size

Value size indique le nombre d'octets contenus dans la valeur de données.

12.17.4.4.2.18 Data value

Data value est la valeur de données qu'il est souhaité d'écrire dans l'objet serveur, l'attribut et l'indice d'attribut identifiés.

12.17.4.4.3 Result

12.17.4.4.3.1 Service contract identifier

Voir 12.17.3.2.2.1.

12.17.4.4.3.2 Priority

Voir 12.17.3.2.2.2.

12.17.4.4.3.3 Discard eligible

Voir 12.17.3.2.2.3.

12.17.4.4.3.4 End-to-end transmission time

Voir 12.17.3.2.2.4.

12.17.4.4.3.5 Forward congestion notification

Voir 12.17.4.3.2.5.

12.17.4.4.3.6 Forward congestion notification echo

Voir 12.17.4.3.3.6.

12.17.4.4.3.7 Server IPv6Address

Voir 12.17.4.3.3.7.

12.17.4.4.3.8 Server TDSAP

Voir 12.17.4.3.2.7.

12.17.4.4.3.9 Server T-port

Voir 12.17.4.3.2.6.

12.17.4.4.3.10 Server object identifier

Ce paramètre identifie un objet serveur dans lequel il est souhaité écrire des données.

12.17.4.4.3.11 Client/source T-port

Ce paramètre identifie l'UAP client associé au port T.

12.17.4.4.3.12 Client TDSAP

Ce paramètre identifie le TDSAP client associé au port T. L'UAP contient l'objet qui est à l'origine de la demande.

12.17.4.4.3.13 Client object identifier

Voir 12.17.4.3.2.12.

12.17.4.4.3.14 Application request identifier

Voir 12.17.4.3.3.14.

12.17.4.4.3.15 Service feedback code

Voir 12.17.4.3.3.16.

12.17.4.5 Execute

12.17.4.5.1 Généralités

Le service Execute est utilisé pour exécuter sur un objet une méthode visible du réseau.

NOTE L'utilisation du service Execute pour établir une méthode de rappel est un moyen de fournir à un serveur le temps approprié pour un message retardé, fournissant des informations en retour au client par l'intermédiaire d'un rappel, plutôt que devoir fournir des résultats d'exécution en temps utile dans la réponse.

Le Tableau 277 définit les primitives du service Execute.

Tableau 277 – Service Execute

Nom du paramètre	Demande	Indication	Réponse	Confirmation
Argument	M	M	-	-
Service contract identifier	M	-	-	-
Priority	M	-	-	-
Discard eligible	M	-	-	-
End-to-end transmission time	-	M(=)	-	-
Forward congestion notification	-	M	-	-
Server T-port	M	-	-	-
Server TDSAP	-	M	-	-
Server object identifier	M	M(=)	-	-
Client IPv6Address	-	M	-	-
Client TDSAP	M	-	-	-
Client T-port	-	M	-	-
Client object identifier	M	M(=)	-	-
Application request ID	M	M(=)	-	-
Method to execute	M	M(=)	-	-
Method identifier	M	M(=)	-	-
Size of input parameters	M	M(=)	-	-
Input parameters	C	C(=)	-	-
Result	-	-	M	M
Service contract identifier	-	-	M	-
Priority	-	-	M	-
Discard eligible	-	-	M	-
End-to-end transmission time	-	-	-	M
Forward congestion notification	-	-	-	M
Forward congestion notification echo	-	-	M	M(=)
Server IPv6Address	-	-	-	M
Server TDSAP	-	-	M	-
Server T-port	-	-	-	M
Server object identifier	-	-	M	M(=)
Client T-port	-	-	M	-
Client TDSAP	-	-	-	M
Client object identifier	-	-	M	M(=)
Application request ID	-	-	M	M(=)
Execution result	-	-	M	M(=)
Service feedback code	-	-	M	M(=)
Size of output parameters	-	-	M	M(=)
Output parameters	-	-	C	C(=)

12.17.4.5.2 Argument

12.17.4.5.2.1 Service contract identifier

Voir 12.17.3.2.2.1.

12.17.4.5.2.2 Priority

Voir 12.17.3.2.2.2.

12.17.4.5.2.3 Discard eligible

Voir 12.17.3.2.2.3.

12.17.4.5.2.4 End-to-end transmission time

Voir 12.17.3.2.2.4.

12.17.4.5.2.5 Forward congestion notification

Voir 12.17.4.3.2.5.

12.17.4.5.2.6 Server T-port

Voir 12.17.4.3.2.6.

12.17.4.5.2.7 Server TDSAP

Voir 12.17.4.3.2.7.

12.17.4.5.2.8 Server object identifier

Voir 12.17.4.3.2.8.

12.17.4.5.2.9 Client/source address

Voir 12.17.4.3.2.9

12.17.4.5.2.10 Client/source TDSAP

Voir 12.17.4.3.2.10.

12.17.4.5.2.11 Client T-port

Voir 12.17.4.3.2.11.

12.17.4.5.2.12 Client object identifier

Voir 12.17.4.3.2.12.

12.17.4.5.2.13 Application request identifier

Voir 12.17.4.3.2.13.

12.17.4.5.2.14 Method identifier

Ce paramètre identifie une méthode de l'objet serveur qu'il est souhaité d'exécuter.

12.17.4.5.2.15 Size of input parameters

Size of input parameters indique le nombre d'octets contenus dans les paramètres d'entrée.

NOTE Les demandes de service Execute et les réponses comprennent la taille en octets du flux de paramètres contenu pour permettre l'analyse (cela est notamment utile dans les scénarios de concaténation d'APDU).

12.17.4.5.2.16 Input parameters

La chaîne de paramètres d'entrée est une chaîne d'octets qui contient les paramètres d'entrée pour la méthode qu'il est demandé d'exécuter. Elle est présente si et seulement si size of input parameters est présent et a une valeur supérieure à zéro.

12.17.4.5.3 Result

12.17.4.5.3.1 Service contract identifier

Voir 12.17.3.2.2.1.

12.17.4.5.3.2 Priority

Voir 12.17.3.2.2.2.

12.17.4.5.3.3 Discard eligible

Voir 12.17.3.2.2.3.

12.17.4.5.3.4 End-to-end transmission time

Voir 12.17.3.2.2.4.

12.17.4.5.3.5 Forward congestion notification

Voir 12.17.4.3.2.5.

12.17.4.5.3.6 Forward congestion notification echo

Voir 12.17.4.3.3.6.

12.17.4.5.3.7 Server IPv6Address

Voir 12.17.4.3.3.7.

12.17.4.5.3.8 Server TDSAP

Voir 12.17.4.3.2.7.

12.17.4.5.3.9 Server T-port

Voir 12.17.4.3.2.6.

12.17.4.5.3.10 Server object identifier

Voir 12.17.4.4.3.10.

12.17.4.5.3.11 Client/source T-port

Voir 12.17.4.4.3.11.

12.17.4.5.3.12 Client TDSAP

Voir 12.17.4.4.3.12.

12.17.4.5.3.13 Client object identifier

Voir 12.17.4.3.2.12.

12.17.4.5.3.14 Application request identifier

Voir 12.17.4.3.3.14.

12.17.4.5.3.15 Execution result

Il contient le résultat de la demande de service d'exécution de la méthode.

12.17.4.5.3.16 Service feedback code

Le service feedback code indique si l'exécution de méthode correspondante a réussi ou pas. Si elle n'a pas réussi, il fournit des informations indiquant pourquoi elle n'a pas réussi.

12.17.4.5.3.17 Size of output parameters

Size of output parameters indique le nombre d'octets contenus dans les paramètres de sortie.

12.17.4.5.3.18 Output parameters

La chaîne de paramètres de sortie est une chaîne d'octets qui contient les paramètres de sortie pour la méthode qui a été exécutée. Elle est présente si et seulement si size of output parameters est présent et a une valeur supérieure à zéro.

12.17.5 Messages unidirectionnels placés en file d'attente acycliques non programmés (source/puits)

12.17.5.1 Généralités

La messagerie unidirectionnelle placée en file d'attente acyclique non programmée est également appelée parfois messagerie source/puits. Ce type d'interaction est utilisé pour les alertes. Les messages envoyés en utilisant ce protocole sont mis en file d'attente par les couches de communication inférieures en vue de l'émission. La réception des messages est non confirmée. Il n'y a ni contrôle de flux ou de débit du processus d'application ni détection de message perdu pour ce mode d'interaction. Comme les communications client/serveur, ces communications exigent l'utilisation d'un contrat de communication, et spécifient la priorité de message sur une base par message.

La largeur de bande pour des communications source/puits n'est pas considérée comme dédiée, mais plutôt comme provenant de la largeur de bande non dédiée (partagée).

Les interactions unidirectionnelles acycliques non programmées dans la présente norme prennent en charge la distribution de messages placés en file d'attente un à un à la demande. Des rapports d'alertes pour la communication de réseau sont toujours émis à partir d'un initiateur, l'ARMO, et sont toujours envoyés vers un seul type de destinataire de message, un objet récepteur d'alerte (ARO).

L'acquittement de la réception d'une alerte peut seulement être issu d'un destinataire de rapport d'alerte (ARO), et est envoyé vers l'objet (ARMO) qui a rapporté l'alerte.

Les services suivants sont fournis pour prendre en charge les communications de messages unidirectionnels placés en file d'attente acycliques non programmés.

- AlertReport;
- AlertAcknowledge; et
- Tunnel.

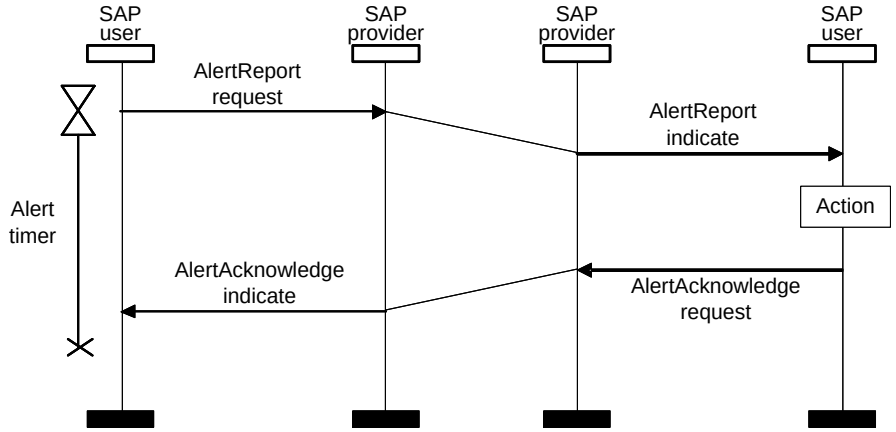
Le service Tunnel est inclus comme un service source/puits pour qu'il puisse tirer profit des capacités de multidiffusion dans le futur. Un tel développement potentiel est un sujet pour une normalisation future.

12.17.5.2 Service AlertReport

12.17.5.2.1 Généralités

AlertReport est utilisé pour rapporter une alerte en utilisant des services de communication unidirectionnels placés en file d'attente. Le contenu du rapport d'alerte dépend du type d'alerte rapporté et de la catégorie de l'alerte. Les AlertReport peuvent être répétés tant qu'un AlertAcknowledge pour l'AlertReport n'a pas été reçu.

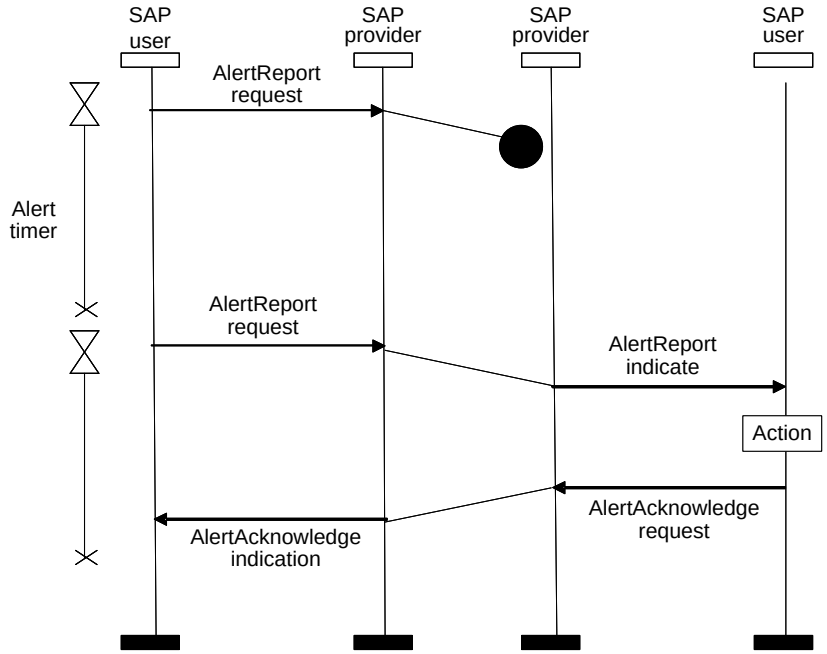
La Figure 130, la Figure 131 et la Figure 132 indiquent l'ordonnancement des messages de rapports d'alertes.



Légende

Anglais	Français
SAP user	Utilisateur de SAP
SAP provider	Fournisseur SAP
AlertReport request	Demande d'AlertReport
AlertReport indicate	Indication d'AlertReport
Action	Action
Alert timer	Temporisateur d'alerte
AlertAcknowledge indicate	Indication d'AlertAcknowledge
AlertAcknowledge request	Demande d'AlertAcknowledge

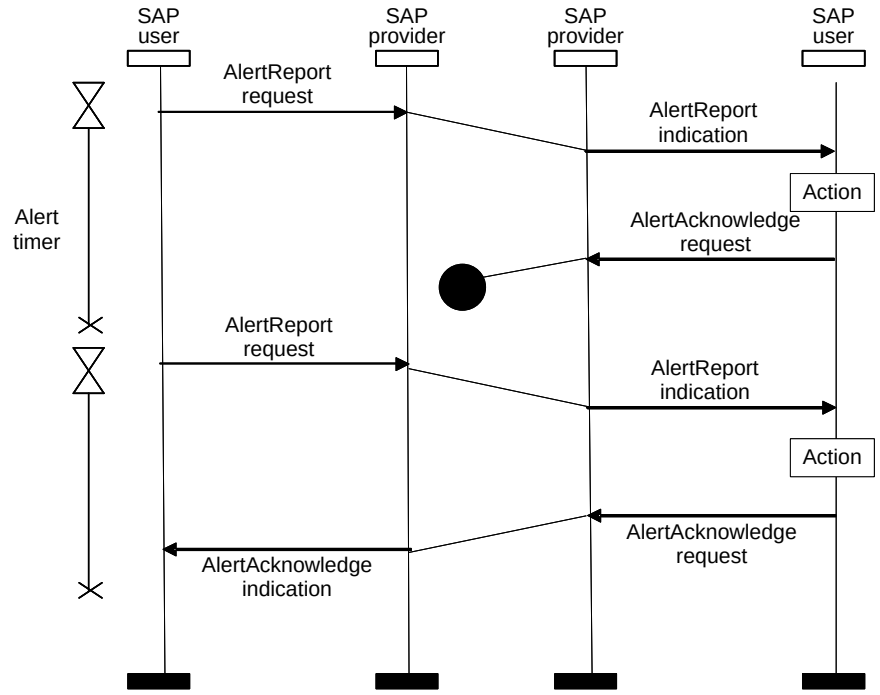
Figure 130 – AlertReport et AlertAcknowledge, livraison réussie



Légende

Anglais	Français
SAP user	Utilisateur de SAP
SAP provider	Fournisseur SAP
Alert timer	Temporisateur d'alerte
AlertReport request	Demande d'AlertReport
AlertReport indicate	Indication d'AlertReport
Action	Action
AlertAcknowlegde indication	Indication d'AlertAcknowlegde
AlertAcknowlegde request	Demande d'AlertAcknowlegde

Figure 131 – AlertReport, échec de livraison



Légende

Anglais	Français
SAP user	Utilisateur de SAP
SAP provider	Fournisseur SAP
AlertReport request	Demande d'AlertReport
AlertReport indication	Indication d'AlertReport
Action	Action
AlertAcknowlegde request	Demande d'AlertAcknowlegde
Alert timer	Temporisateur d'alerte
AlertAcknowlegde indication	Indication d'AlertAcknowlegde

Figure 132 – AlertReport, échec d'acquittement

NOTE 1 La politique de temporisation/répétition des AlertReport est définie sur l'ARMO. Voir 6.2.7.2 pour plus de détails.

La production de rapports d'alerte utilise deux services distincts de communication d'application à deux parties. Pour rapporter une alerte, l'AP utilise le service AlertReport. Dans la présente version de la norme, le destinataire d'un AlertReport doit acquitter l'AlertReport en utilisant le service AlertAcknowledge.

NOTE 2 L'utilisation de deux services distincts, AlertReport et AlertAcknowledgement, permet qu'une seule et même alerte soit envoyée vers plusieurs destinations dans une future révision à la présente norme.

La surveillance et la vérification pour détecter un acquittement, ainsi que la production à nouveau d'un rapport relatif à la condition d'alerte pour laquelle un acquittement n'a pas été reçu relèvent de la responsabilité de l'appareil rapporteur d'alerte. La production à nouveau d'un rapport relatif à une alerte qui n'est plus actuelle, et pour laquelle une indication d'AlertAcknowledge n'a pas été reçue, relève d'une initiative locale. Par exemple, si une situation de diagnostic se produit et une alerte est rapportée, et ensuite l'appareil rapporteur se réamorce de telle manière que la situation de diagnostic ne soit plus actuelle, l'appareil pourrait ne pas rapporter à nouveau l'alerte de diagnostic qui était en vigueur avant le réamorçage bien qu'aucun AlertAcknowledge n'ait été reçu.

AlertReport utilise le même modèle de communication comme étant une primitive client/serveur à deux parties. Le Tableau 278 définit les primitives de service pour le service AlertReport.

Tableau 278 – Service AlertReport

Nom du paramètre	Demande	Indication
Argument	M	M
Service contract identifier	M	-
Priority	M	-
Discard eligible	M	-
End-to-end transmission time	-	M
ARMO TDSAP	M	-
ARMO T-port	-	M
ARMO	M	M(=)
Sink T-port	M	-
Sink TDSAP	-	M
Sink object identifier	M	M(=)
Individual alert report	M	M(=)
Individual alert identifier	M	M(=)
Alert detector T-port	M	M(=)
Alert detector object	M	M(=)
Detection time	M	M(=)
Alert class	M	M(=)
Alarm direction	C	C(=)
Alert category	M	M(=)
Alert priority	M	M(=)
Type d'alerte	M	M(=)
Associated-data size	M	M(=)
Associated data	O	O(=)

12.17.5.2.2 Argument

12.17.5.2.2.1 Service contract identifier

Voir 12.17.3.2.2.1.

12.17.5.2.2.2 Priority

Voir 12.17.3.2.2.2.

12.17.5.2.2.3 Discard eligible

Voir 12.17.3.2.2.3.

12.17.5.2.2.4 End-to-end transmission time

Voir 12.17.3.2.2.4.

12.17.5.2.2.5 Alert reporting management object TDSAP

Ce paramètre indique le TDSAP de l'application qui émet ce rapport d'alerte.

12.17.5.2.2.6 Alert reporting management object T-port

Ce paramètre indique le port T de l'application qui émet ce rapport d'alerte.

12.17.5.2.2.7 Alert reporting management object

Ce paramètre représente l'identificateur d'objet de l'ARMO qui rapporte l'alerte.

12.17.5.2.2.8 Sink T-port

Ce paramètre identifie l'UAP de puits associé au port T.

12.17.5.2.2.9 Sink TL data service access point

Ce paramètre indique le TDSAP correspondant au puits du port T.

12.17.5.2.2.10 Sink object identifier

Ce paramètre spécifie l'objet puits de destination dans l'application vers laquelle cette demande de service doit être envoyée.

12.17.5.2.2.11 Alert source IPv6Address

Ce paramètre identifie l'adresse IPv6Address de la source de cette demande.

12.17.5.2.2.12 Alert source TDSAP

Ce paramètre identifie le TDSAP associé de l'UAP source.

12.17.5.2.2.13 Source T-port

Ce paramètre identifie l'UAP source d'application émetteur associé au port T.

12.17.5.2.2.14 Individual alert report

Ce paramètre contient une alerte individuelle rapportée par cette invocation de service.

12.17.5.2.2.15 Individual alert identifier

Ce paramètre identifie de façon univoque le rapport d'alerte individuelle. Des séquences distinctes de valeurs d'identificateur pour les catégories de rapports d'alertes doivent être maintenues. La valeur de ce paramètre doit augmenter de façon monotone et doit repasser par zéro lorsque la valeur maximale est atteinte. Elle est incluse lorsqu'un rapport d'alerte individuelle est acquitté. Elle est également utilisée par un récepteur d'alerte pour déterminer si un rapport d'alertes ou un ensemble de rapports d'alertes d'une catégorie particulière a été manqué. Si une condition de rapport manqué est détectée, il convient d'accomplir une opération de récupération d'alarme. Se référer aux attributs Alarm_Regen de l'ARMO en 6.2.7.2 pour plus de détails relatifs au déclenchement de la régénération.

12.17.5.2.2.16 Alert source transport port

Ce paramètre identifie l'UAP contenant l'objet qui a détecté l'alerte par l'intermédiaire de son port T associé.

12.17.5.2.2.17 Alert source object

L'objet source d'alerte indique l'instance d'objet qui a détecté la condition d'alarme.

NOTE L'objet gestion de rapports d'alertes rapporte les conditions d'alerte détectées par un ou plusieurs objets détecteurs d'alerte.

12.17.5.2.2.18 Detection time

Ce paramètre spécifie le temps auquel la condition d'alerte a été détectée. Cette valeur indique le temps réseau auquel la condition d'alerte a été détectée. La façon de rendre les informations de temps disponibles à une application rapportant une alerte est une question locale d'appareil, non spécifiée par la la présente norme.

NOTE La conversion du temps réseau en temps social (temps d'horloge murale), lorsque souhaitée, est accomplie dans la passerelle. Voir 5.6 pour plus de détails.

12.17.5.2.2.19 Alert class

Ce paramètre indique s'il s'agit d'un type d'alerte événement (sans état) ou alarme (orientée état).

12.17.5.2.2.20 Alarm direction

Pour les alertes qui sont orientées état (les alarmes), il indique si le rapport concerne une condition d'alarme ou un retour à la normale à partir d'une condition d'alarme.

12.17.5.2.2.21 Alert category

La catégorie d'alertes indique si l'alerte est une alerte de diagnostics d'appareil, une alerte de diagnostics de communication, une alerte de sécurité ou une alerte de processus.

12.17.5.2.2.22 Alert priority

La priorité d'alerte est une valeur qui suggère l'importance de l'alerte. Plus la valeur est grande, plus l'alerte est importante. Les systèmes d'hôte mettent en correspondance les priorités d'appareil aux priorités d'alerte d'hôte qui comprennent habituellement les priorités "urgent", "high", "medium", "low" et "journal". Le mapping recommandé des valeurs de priorité d'alerte en ces catégories est comme suit:

- 0..2: journal
- 3..5: low
- 6..8: medium
- 9..11: high
- 12..15: urgent

Sachant que l'interprétation des priorités d'alerte se produit principalement dans les appareils initiateur et récepteur prévus, d'autres assignations qui reflètent une catégorisation différente sont permises.

12.17.5.2.2.23 Type d'alerte

Le type d'alerte fournit des informations complémentaires relatives à l'alerte, spécifiques à la catégorie d'alerte).

12.17.5.2.2.24 Associated-data size

Associated-data size spécifie la taille de toutes données spécifiques à une alerte acheminées avec l'alerte.

12.17.5.2.2.25 Associated data

Associated data fournit un moyen d'acheminer des données spécifiques à une alerte

12.17.5.3 Service AlertAcknowledge

12.17.5.3.1 Généralités

AlertAcknowledge est un service à deux parties qui est utilisé pour acquitter une alerte individuelle auprès d'un objet de gestion de rapports d'alertes. Pour les rapports d'alerte en monodiffusion, la réception d'un AlertAcknowledge doit entraîner la cessation des demandes de répétitions de tentative d'AlertReport pour l'alerte individuelle correspondante.

Un AlertAcknowledge doit être envoyé pour chaque AlertReport reçu.

NOTE Si un AlertReport double a été reçu, soit l'application qui a envoyé l'AlertReport n'a pas reçu l'AlertAcknowledge dans sa temporisation/temps de répétitions de tentative, soit le message d'AlertAcknowledgement n'a pas été reçu. Etant donné que l'application envoyant l'AlertAcknowledge ne sait pas quelle situation s'est produite, un doublon d'acquittement est envoyé.

Le service AlertAcknowledge est décrit dans le Tableau 279.

Tableau 279 – Service AlertAcknowledge

Nom du paramètre	Demande	Indication
Argument	M	M
Service contract identifier	M	-
Priority	M	-
Discard eligible	M	-
End-to-end transmission time	-	M
Source IPv6Address	-	M
Source TDSAP	M	-
Source T-port	-	M
Source object identifier	M	M(=)
Destination transport port	M	-
Destination TDSAP	-	M
Destination object identifier	M	M(=)
Individual alert identifier	M	M(=)

12.17.5.3.2 Argument

12.17.5.3.2.1 Service contract identifier

Voir 12.17.3.2.2.1.

12.17.5.3.2.2 Priority

Voir 12.17.3.2.2.2.

12.17.5.3.2.3 Discard eligible

Voir 12.17.3.2.2.3.

12.17.5.3.2.4 End-to-end transmission time

Voir 12.17.3.2.2.4.

12.17.5.3.2.5 Source IPv6Address

Ce paramètre identifie l'adresse IPv6Address pour la source de cette demande.

12.17.5.3.2.6 Source TDSAP

Ce paramètre identifie le TDSAO associé de l'AP source de primitive de service. Le TDSAP établit un mapping 1:1 à un UAP.

12.17.5.3.2.7 Source T-port

Ce paramètre identifie l'UAP source d'application émetteur associé au port T.

12.17.5.3.2.8 Source object identifier

Ce paramètre identifie l'objet qui initie l'acquittement d'alerte.

12.17.5.3.2.9 Destination T-port

Ce paramètre identifie le processus d'application pour recevoir l'acquittement d'alerte, car une seule application est associée à un port T.

12.17.5.3.2.10 Destination TDSAP

Ce paramètre identifie le SAP de TDLE correspondant au port de transport de destination.

12.17.5.3.2.11 Destination object identifier

Ce paramètre identifie l'objet dans le processus d'application de l'appareil pour recevoir l'acquittement.

12.17.5.3.2.12 Individual alert identifier

Ce paramètre identifie l'alerte individuelle qui est acquittée.

12.17.6 Aspects communs client/serveur et à la source/puits

12.17.6.1 Messagerie individuelle ou concaténée pour client/serveur et/ou source/puits

Les messages client/serveur et source/puits peuvent être envoyés sous la forme d'une unité de données de service de transport (TSDU) individuelle ou peuvent être concaténés ensemble au sein d'une seule TSDU. La concaténation prend en charge les messages de primitives tant à quatre parties ("request" et "response") qu'à deux parties ("request" seulement). La concaténation permet que des messages client/serveur soient combinés dans la TSDU avec des messages de source/puits. La façon de déterminer et d'accomplir la concaténation des APDU est une question locale d'appareil. Il convient que les concaténations s'abstiennent d'inclure plus de deux services, car cela peut se traduire par une communication avec plus de salves.

NOTE 1 Le débat relatif à la violation d'acquittement en laps de temps de l'IETF RFC 2525 fournit le contexte de la concaténation d'acquittements de message et les ramifications de la présence de plus deux tels acquittements dans une PDU.

NOTE 2 La P/S (éditer/s'abonner) prend déjà en charge l'inclusion de plusieurs valeurs issues d'un UAP dans un seul message. Voir 12.15.2.5 et 12.15.2.6 pour de plus amples détails.

La concaténation peut être utilisée pour réduire le surdébit d'émission et/ou livrer un jeu de messages à l'ASL correspondante comme un bloc.

Tous les messages au sein de la concaténation doivent avoir une priorité de message commune, et doivent indiquer la communication par l'intermédiaire d'un contrat de communication commun.

Le nombre de services d'ASL qui peuvent être concaténés par une ASL construisant la PDU concaténée est limité par la taille d'APDU maximale correspondant au contrat de communication devant être utilisé pour les messages.

L'ASL recevant une APDU concaténée doit analyser et traiter chaque APDU individuellement du début à la fin jusqu'à ce que la fin de la TSDU ait été atteinte. Si une erreur de protocole est détectée pendant l'analyse d'une APDU concaténée, une seule erreur d'APDU mal formée est indiquée et la partie restante de l'APDU doit être rejetée.

Une concaténation de services d'ASL peut contenir:

- des primitives de service d'ASL homogènes (par exemple, toutes les demandes de service); ou
- des primitives de service d'ASL hétérogènes (par exemple: flux client/serveur, demandes et réponses; peuvent être mélangés);
- des primitives de flux de communication d'application homogènes (toutes les client/serveur, par exemple); ou
- des primitives de types de flux de communication d'application hétérogènes (par exemple: client/serveur et source/puits).

L'AL elle-même ne donne aucune exigence sur la façon dont des réponses aux services inclus dans une concaténation sont retournées; la détermination de celle-ci est laissée à la discrétion de l'application réceptrice. Par exemple, si une demande client/serveur concaténée contient les demandes de service (A, B, C) et une autre demande de service client/serveur concaténée contient les demandes de service (D, E), les réponses client/serveur à celles-ci peuvent:

– ne pas être exigées (par exemple: A, B, et D peuvent être des services à quatre parties qui exigent une réponse, mais C et E peuvent être des services à deux parties qui n'exigent pas de réponse);
– ne pas être concaténées du tout et être retournées dans un ordre quelconque (par exemple, réponse A, réponse B, réponse E, réponse D, réponse C, toutes dans des APDU distinctes);
– être partiellement concaténées, dans un ordre quelconque (par exemple: la réponse retournée peut être B, C dans une APDU, A dans une autre APDU);
– utiliser une seule APDU pour répondre à une demande concaténée, mais les réponses peuvent être concaténées dans n'importe quel ordre (par exemple: la réponse retournée peut être concaténée sous la forme BCA);
– être complètement concaténées dans le même ordre (par exemple, la réponse est retournée en étant concaténée sous la forme ABC);
– être complètement concaténées différemment des demandes reçues (par exemple, la réponse peut être retournée sous la forme BD, ACE).

La façon et le moment de l'initiation d'une communication par une ASL déterminent le moment auquel créer une concaténation ainsi que le moment auquel livrer la concaténation à la suite inférieure de protocoles de communication en vue de l'acheminement; cela relève d'une initiative locale. Cependant, la présente norme doit spécifier la structure globale d'une TSDU contenant des APDU concaténées.

La présente norme ne donne pas d'exigence relative à l'ordre dans lequel les services inclus dans la messagerie concaténée sont traités par la destination. Ainsi l'ordre des réponses n'est pas tenu d'être le même que l'ordre des demandes incluses dans la concaténation.

NOTE 3 Une mise en œuvre qui définit des services locaux pour encadrer les constructions concaténées peut fournir un contrôle supplémentaire sur le contenu de concaténation.

La Figure 133 montre une réponse concaténée pour plusieurs demandes d'écriture en cours sans perte de message. Des temporisations sont décrites en 12.12.4.2.2.1.

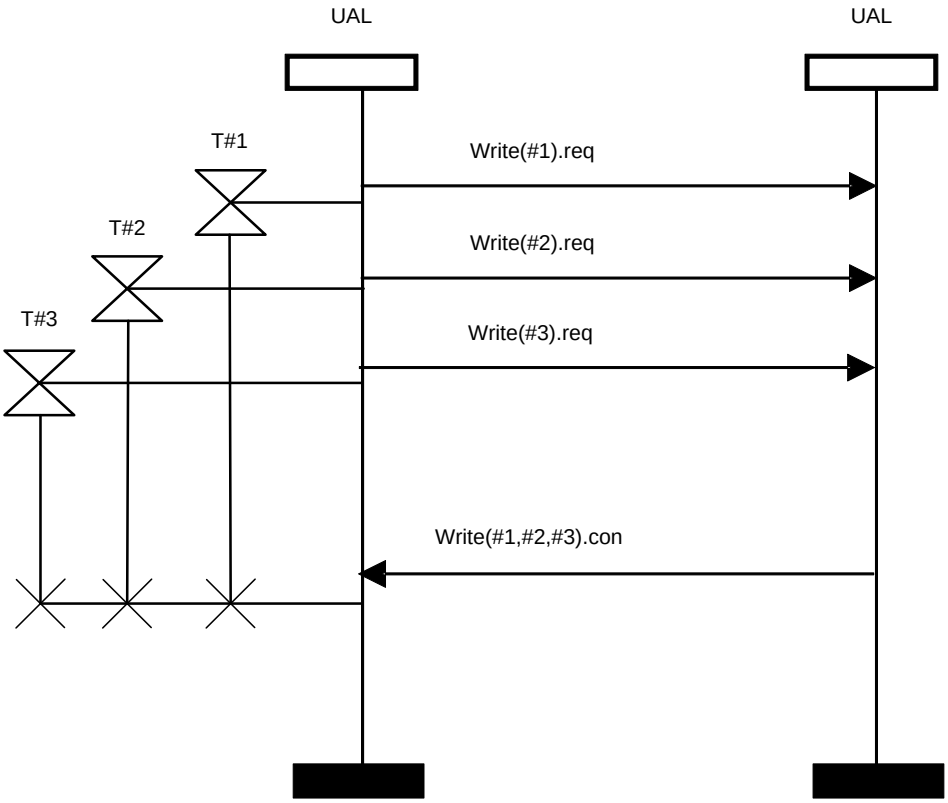


Figure 133 – Réponse concaténée pour plusieurs demandes d'écriture en cours (pas de perte de message)

12.17.6.2 Services communs de sous-couche d'application pour la messagerie client/serveur et source/puits – Tunnel

12.17.6.2.1 Généralités

Le service tunnel peut utiliser un modèle de primitives de service de communication soit à deux parties (source/puits), soit à quatre parties (client/serveur).

La responsabilité d'inclure au sein de la charge utile non native de messages tunnel les informations pour permettre la concordance de primitives de service, l'accomplissement du traitement approprié au niveau application des situations d'erreur telles que la détection de doublons de message et la détection d'une livraison dans le désordre, etc., doit incomber au processus d'application.

Le service tunnel peut être utilisé pour encapsuler les communications tant client/serveur que source/puits telles que définies par la présente norme. Ce service identifie un message comme étant destiné à un tunnel de protocoles non natifs. L'identificateur de service est exigé pour que l'ASL puisse analyser un message et déterminer de le passer à un tunnel de protocoles hérités ou de le traiter comme étant un message natif. Le service tunnel fournit un seul niveau d'encapsulation de message pour un tunnel de protocoles. L'APDU non native est passée à l'objet de destination spécifié dans la demande de service tunnel.

Une application de tunnellisation peut établir la politique de répétitions de tentative pour des demandes à deux parties (source/puits) qu'elle envoie.

Le Tableau 280 définit les primitives du service Tunnel.

Tableau 280 – Service Tunnel

Nom du paramètre	Demande	Indication	Réponse	Confirmation
Argument	M	M	-	-
Service contract identifier	M	-	-	-
Priority	M	-	-	-
Discard eligible	M	-	-	-
End-to-end transmission time	-	M	-	-
Forward congestion notification	-	M	-	-
Application destination T-port	M	-	-	-
Application destination TDLE SAP	-	M	-	-
Application destination object identifier	M	M(=)	-	-
Application source IPv6Address	-	M	-	-
Application source TLDE SAP	M	-	-	-
Application source T-port	-	M	-	-
Application source object identifier	M	M(=)	-	-
Payload size (in octets)	M	M(=)	-	-
Tunnel payload data	M	M(=)	-	-
Result	-	-	U	U
Service contract identifier	-	-	M	-
Priority	-	-	M	-
Discard eligible	-	-	M	-
End-to-end transmission time	-	-	-	M
Forward congestion notification	-	-	-	M
Forward congestion notification echo	-	-	M	M(=)
Application destination T-port	-	-	M	-
Application destination TDLE SAP	-	-	-	M
Application destination object identifier	-	-	M	M(=)
Application source IPv6Address	-	-	-	M
Application source TLDE SAP	-	-	M	-
Application source T-port	-	-	-	M(=)
Application source object identifier	-	-	M	M(=)
Payload size (in octets)	-	-	M	M(=)
Tunnel payload data	-	-	M	M(=)

12.17.6.2.2 Argument

12.17.6.2.2.1 Service contract identifier

Voir 12.17.3.2.2.1.

12.17.6.2.2.2 Priority

Voir 12.17.3.2.2.2.

12.17.6.2.2.3 Discard eligible

Voir 12.17.3.2.2.3.

12.17.6.2.2.4 End-to-end transmission time

Voir 12.17.3.2.2.4.

12.17.6.2.2.5 Forward congestion notification

Ce paramètre indique si la demande a rencontré un encombrement de réseau sur son chemin allant du client vers le serveur.

12.17.6.2.2.6 Application destination T-port

Ce paramètre identifie l'UAP à la destination pour cette demande de service.

12.17.6.2.2.7 Application destination TDSAP

Ce paramètre représente le TDSAP correspondant au port de destination de transport d'AL.

12.17.6.2.2.8 Application destination object identifier

Ce paramètre identifie l'objet dans l'application réceptrice.

12.17.6.2.2.9 Application source IPv6Address

Ce paramètre identifie l'adresse IPv6Address pour le client de cette demande.

12.17.6.2.2.10 Application source TDSAP

Ce paramètre identifie le TDSAP associé de l'UAP source d'application. Le TDSAP établit un mapping 1:1 à un UAP. L'UAP contient l'objet à l'origine de demande dans le client.

12.17.6.2.2.11 Application source T-port

Ce paramètre identifie l'UAP qui est la source pour cette demande de service.

12.17.6.2.2.12 Application source object identifier

Ce paramètre indique l'objet source d'application qui a initié la demande de service Tunnel.

12.17.6.2.2.13 Payload size

Ce paramètre indique le nombre d'octets du paramètre de charge utile de tunnel.

12.17.6.2.2.14 Tunnel payload data

Ce paramètre représente les données (APDU de protocole hérité, par exemple) qui doivent être acheminées vers l'objet serveur.

12.17.6.2.3 Result

12.17.6.2.3.1 Service contract identifier

Voir 12.17.3.2.2.1.

12.17.6.2.3.2 Priority

Voir 12.17.3.2.2.2.

12.17.6.2.3.3 Discard eligible

Voir 12.17.3.2.2.3.

12.17.6.2.3.4 End-to-end transmission time

Voir 12.17.3.2.2.4.

12.17.6.2.3.5 Forward congestion notification

Ce paramètre indique si la réponse de service a rencontré un encombrement de réseau sur son chemin allant du serveur vers le client.

12.17.6.2.3.6 Forward congestion notification echo

Ce paramètre indique si la demande de service a rencontré un encombrement de réseau sur son chemin allant du client vers le serveur.

12.17.6.2.3.7 Application destination T-port

Ce paramètre identifie l'UAP à la destination pour cette demande de service.

12.17.6.2.3.8 Application destination TDSAP

Ce paramètre représente le TDSAP correspondant au port de destination de transport d'AL.

12.17.6.2.3.9 Application destination object identifier

Ce paramètre indique l'objet de destination d'application qui a initié la demande de service Tunnel.

12.17.6.2.3.10 Application source IPv6Address

Ce paramètre identifie l'adresse IPv6Address pour le client de cette demande.

12.17.6.2.3.11 Application source TDSAP

Ce paramètre identifie le TDSAP associé de l'UAP source d'application. Le TDSAP établit un mapping 1:1 à un UAP. L'UAP contient l'objet à l'origine de demande dans le client.

12.17.6.2.3.12 Application source T-port

Ce paramètre indique l'objet source d'application qui a initié la demande de service Tunnel.

12.17.6.2.3.13 Application source object identifier

Ce paramètre indique l'objet tunnel source d'application qui a initié la demande de service Tunnel.

12.17.6.2.3.14 Payload size

Ce paramètre indique le nombre d'octets du paramètre de charge utile de tunnel.

12.17.6.2.3.15 Tunnel payload data

Ce paramètre représente les données (APDU de protocole hérité, par exemple) qui doivent être acheminées vers l'objet serveur.

12.18 Utilisation du flux d'AL relative aux services de couche inférieure

12.18.1 Généralités

Tous les types de messagerie (par exemple, publication, client/serveur, source/puits, transfert en masse) et toutes les qualités de service peuvent circuler par le TDSAP commun pour le processus d'application. Le Tableau 281 indique le mapping des flux d'AL aux services de TL fournis.

Tableau 281 – Caractéristiques des flux d'application

Type de flux d'application	Placé en tampon ou placé en file d'attente	Périodique (programmé)	Sensible à l'ordre	Fiabilité	Non acquitté	Importance du message		
						Elevée	Moyenne	Faible
Edition/abonnement périodique	En tampon	Oui	Oui	Non	Oui	[a]	[a]	[a]
Client/serveur	En file d'attente	Non	Non	Non	Oui	[a]	[a]	[a]
Source/puits	En file d'attente	Non	Non	Non	Oui	[a]	[b]	[a]

[a] Parmi les alternatives d'importance de message, celle qui est sélectionnée s'applique.

[b] Aucun cas d'utilisation identifié.

12.18.2 Utilisation d'AL des TDSAP

L'ASL communique avec les couches inférieures de la suite de protocoles de communication par l'intermédiaire des TDSAP. L'information communiquée à la TL pour le mapping de TDSAP est un sous-ensemble des informations communiquées à l'ASL à partir de l'UAP au niveau de l'ASAP. Il y a un TDSAP bien connu dans la présente norme, à savoir le TDSAP numéro 0, qui est utilisé pour des communications avec les objets représentés par l'application DMAP.

Les TDSAP qui ne sont pas bien connus peuvent être associés à un et un seul processus d'application particulier. C'est-à-dire, il y a une relation de mapping univoque (1:1) entre un TDSAP et un processus d'UAL. Les TDSAP étant locaux, les entités distantes indiquent un port T qui représente une application correspondante. Les ports T sont en correspondance univoque (1:1) avec les TDSAP. Les relations processus d'UAL/TDSAP/port de données de transport doivent survivre à un redémarrage d'application.

NOTE Quinze TDSAP sont disponibles pour une émission compressée. Voir 11.6 pour plus de détails.

12.18.3 Mapping des primitives de service d'AL aux primitives de service de TL

Le Tableau 282 indique le mapping de primitives de service d'application à des services de transport.

Tableau 282 – Mapping de primitives de service d'AL à des primitives de service de TL

Primitive de service d'application	Service de transport	Données acheminées entre sous-couche application et service de transport
Publish.request Read.request, Read.response Write.request, Write.response Execute.request, Execute.response Tunnel.request, Tunnel.response AlertReport.request AlertAcknowledge.request	T-DATA.request	Contract_ID APDU_size APDU Message priority Discard eligibility Source TDSAP Destination T-port NOTE 1 L'ID de contrat indique l'adresse de destination et la priorité du contrat. NOTE 2 Le port T source peut être déterminé à partir du TDSAP source, mais il est explicitement passé pour concorder avec l'interface fournie par la TL.
Publish.indicate Read.indicate, Read.confirmation Write.indicate, Write.confirmation Execute.indicate, Execute.confirmation Tunnel.indicate, Tunnel.confirmation AlertReport.indicate AlertAcknowledge.indicate	T-DATA.indicate	Source IPv6Address Source T-port APDU_size (équivalent à TSDU size) APDU (équivalent de TSDU) Explicit congestion notification (ECN) Destination TDSAP Destination T-port Transport time (temps de livraison de bout en bout dans un seul sens, en secondes)

12.19 Gestion d'AL

12.19.1 Généralités

La gestion d'AL prend en charge l'objet local de gestion de sous-couche d'application de DMAP. L'accès aux attributs et aux méthodes de cet objet est défini par l'ASL. Pour la présente norme, l'ASL fournit l'accès pour lire une valeur configurée dans la MIB de l'ASL, pour écrire une valeur configurée dans la MIB de l'ASL, et pour prendre en charge la réinitialisation de l'ASL.

12.19.2 Traitement de sous-application des unités de données de protocole d'application mal formées

L'ASL prend en charge le fait d'informer le DMAP local d'un problème potentiel d'appareil/de communication si un seuil configuré par la gestion d'AL est atteint dans les limites de la période de temps configurée pour les APDU mal formées reçues en provenance d'un appareil source particulier. Quelques exemples d'APDU mal formées sont:

- APDU de taille incorrecte (trop longue ou trop courte);
- identificateur de service non valide; ou
- mauvaise utilisation de service (par exemple, la primitive "response" a été indiquée dans la PDU pour un service client/serveur ou source/puits à deux parties).

L'intention de cette information est de permettre au DMAP de fournir des informations à la gestion de niveau supérieur. Cela peut être important, par exemple, pour permettre de détecter que l'attaque d'une APDU mal formée se produit.

L'ASL peut être configurée pour aviser le DMAP chaque fois qu'une APDU mal formée est reçue.

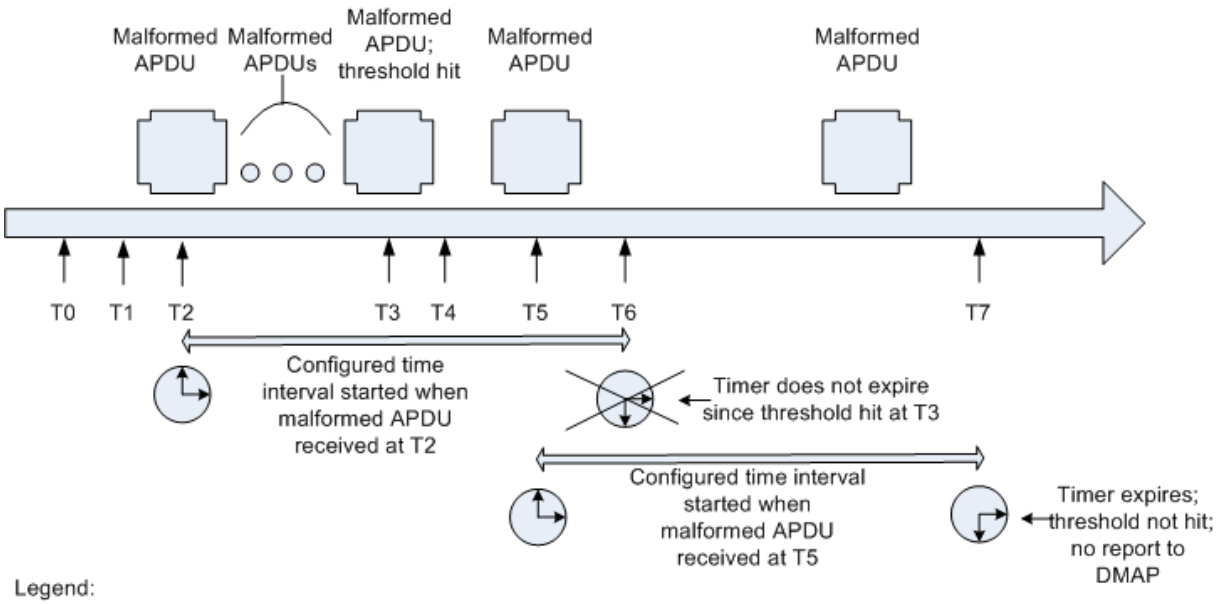
L'ASL peut être configurée avec des valeurs différentes de zéro pour un seuil et un intervalle de temps. Lorsqu'elle est ainsi configurée, l'ASL maintiendra, en interne, des compteurs et temporisateurs individuels pour chaque adresse de source réseau en provenance de laquelle une APDU mal formée a été reçue. Le compte débute avec la réception de la première APDU mal formée en provenance d'un appareil et se poursuit soit tant que la valeur-seuil d'APDU mal formées n'est pas atteinte, soit tant que l'ASL_TimePeriodForMalformedAPDUs n'a pas expiré.

Si la valeur-seuil d'APDU mal formées est atteinte avant ou à l'expiration de l'intervalle de temps configuré, le DMAP est avisé et le décompte et l'intervalle de temps pour l'appareil sont réinitialisés.

Si le seuil d'APDU mal formées n'est pas atteint dans les limites de l'intervalle de temps configuré, les compteurs et les temporisateurs sont intérieurement réinitialisés.

NOTE La façon dont le DMAP est avisé d'une APDU mal formée ou qu'un seuil est atteint dans un intervalle de temps relève d'une initiative locale et n'est donc pas spécifiée par la présente norme.

La Figure 134 montre le traitement des APDU mal formées.



Legend:
T0 – malformed APDU threshold is set to non-zero value
T1 – malformed APDU counting interval is set to non-zero value
T2 – first malformed APDU arrives from source device "X", time interval for counting malformed APDUs starts
T3 – threshold of malformed APDUs is reached
T4 – report to DMAP after which internal ASL internal counter resets
T5 -- another malformed APDU arrives again from source device "X"
T6 - original calculation of expiration of time interval for malformed APDUs from device "X" , starting from receipt of malformed APDU at T2
T7 – time interval started at T5 expires without threshold being hit; ASL internal counter resets

NOTE: The order of performing steps T0 and T1 is not important.

Légende

Anglais	Français
Malformed APDU	APDU mal formée
Malformed APDUs	APDU mal formées
Malformed APDU; threshold hit	APDU mal formée; seuil atteint

Anglais	Français
Configured time interval started when malformed APDU received at T2	L'intervalle de temps configuré a démarré lorsqu'une APDU mal formée a été reçue à T2
Timer does not expire since threshold hit at T3	Le temporisateur n'expire pas, car le seuil est atteint à T3
Configured time interval started when malformed APDU received at T5	L'intervalle de temps configuré a démarré lorsqu'une APDU mal formée a été reçue à T3
Timer expires; threshold not hit; no report to DMAP	Temporisateur expiré; seuil non atteint; pas de rapport à DMAP
Legend:	Légende:
T0- malformed APDU threshold is set to non-zero value	T0 – le seuil d'APDU mal formées est mis à une valeur différente de zéro
T1- malformed APDU counting interval is set to non-zero value	T1 – l'intervalle de comptage d'APDU mal formées est mis à une valeur différente de zéro
T2- first malformed APDU arrives from source device “X”, time interval for counting malformed APDUs starts	T2- la première APDU mal formée arrive de l'appareil source « X », l'intervalle de temps pour compter les APDU mal formées démarre
T3- threshold of malformed APDUs is reached	T3 – le seuil des APDU mal formées est atteint
T4- report to DMAP after which internal ASL internal counter resets	T4 – rapporter au DMAP après quoi le compteur interne d'ASL interne se réinitialise
T5- another of malformed APDU arrives again from source device “X”	T5- une autre APDU mal formée arrive encore de l'appareil source « X »
T6- original calculation of expiration of time interval for malformed APDUs from deviced “X”, starting from malformed APDU at T2	T6- calcul initial de l'expiration d'intervalle de temps pour les APDU mal formées issues de l'appareil « X », en commençant par l'APDU malformée à T2
T7- time interval started at TS expired without threshold being hit; ASL internal counter resets	T7- l'intervalle de temps démarré à T5 a expire sans que le seuil ait été atteint; le compteur interne d'ASL se réinitialise
NOTE The order of performing steps T0 and T1 is not important	NOTE L'ordre d'exécution des étapes T0 et T1 n'est pas important

Figure 134 – Gestion et traitement des APDU mal formées reçues en provenance de l'appareil X

12.19.3 Attributs de l'objet de gestion de sous-couche d'application

Le Tableau 283 décrit les attributs pris en charge par l'objet de gestion de sous-couche d'application (ASLMO).

Tableau 283 – Attributs d'ASLMO *(1 de 2)*

Nom du type d'objet normalisé: Application sublayer management object (ASLMO, objet de gestion de sous-couche d'application)				
Identificateur du type d'objet normalisé: 121				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
ObjectIdentifier	Identificateur-clé d'objet	Identificateur unique pour l'objet	Type: Unsigned16 Classification: Constant Valeur par défaut: 7 Plage valide: 7	N/A
Réservée pour un usage futur	0	-	-	-
MalformedAPDUsAdvise	1	Indique s'il convient que l'ASL indique au DMAP local chaque occurrence d'une APDU mal formée	Type: Boolean Classification: Static Accessibilité: Lecture/écriture Valeur par défaut initiale:false	Si ce paramètre a la valeur True, l'ASL doit passer au DMAP local chaque APDU mal formée qu'elle reçoit
TimeIntervalForCountingMalformedAPDUs	2	Cet attribut spécifie l'intervalle de temps pour que l'ASL compte les APDU mal formées reçues en provenance d'un appareil particulier. Le compte se produit à partir de la détection de la première APDU mal formée provenant d'un appareil. Cet intervalle est communément appliqué aux APDU provenant de toutes les adresses IPv6Addresses. L'unité utilisée pour cet attribut est la seconde	Type: TAITimeDifference Classification: Static Accessibilité: Lecture/écriture Valeur par défaut initiale: 0 Plage valide: 0 ≤ valeur ≤ à 86 400 s Le nombre de jours n'est pas inclus (il est toujours égal à zéro)	Si l'intervalle de temps expire sans que le seuil soit atteint, les informations correspondantes de compteur et de temporisateur de l'ASL mal formée doivent être réinitialisées à zéro (0)
MalformedAPDUsThreshold	3	Valeur-seuil commune à appliquer aux APDU mal formées reçues en provenance de chaque appareil	Type: Unsigned16 Classification: Static Accessibilité: Lecture/écriture Valeur par défaut initiale: 0	Si ce seuil est atteint dans l'intervalle de temps spécifié, une alerte de communication doit être rapportée indiquant l'appareil qui envoyait des APDU mal formées. Si une valeur-seuil est établie alors que le compte est en cours, et la valeur établie est inférieure au seuil antérieur si bien que le nouveau seuil a été dépassé, une alerte d'APDU mal formée doit être rapportée

Tableau 283 *(2 de 2)*

Nom du type d'objet normalisé: Application sublayer management object (ASLMO, objet de gestion de sous-couche d'application)				
Identificateur du type d'objet normalisé: 121				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
MalformedAPDUAlertDescriptor	4	Décrit comment une alerte mal formée est rapportée sur le réseau	Type: Descripteur de rapports d'alertes Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: [true 7]	
MaxDevicesForWhichMalformed APDUsCanBeCounted	5	Décrit la capacité de comptages des APDU mal formées de l'ASL en termes de nombre maximal d'appareils pour lesquels les comptes peuvent être maintenus simultanément	Type: Unsigned16 Classification: Constant Accessibilité: Lecture seule	Une valeur minimale exigée peut être établie par exemple en fonction du rôle de l'appareil
Réservé pour usage futur par la présente norme	6..63	-	-	-

Les attributs classés comme étant "constant" ou "static" doivent être préservés dans les redémarrages et les pannes de courant.

12.19.4 Méthodes de l'objet de gestion de sous-couche d'application

12.19.4.1 Méthodes d'objets normalisés

Un ASLMO a les méthodes définies dans le Tableau 284.

Tableau 284 – Méthodes de l'objet de gestion de sous-couche d'application

Nom du type d'objet normalisé: Application sublayer management object (ASLMO, objet de gestion de sous-couche d'application)		
Identificateur du type d'objet normalisé: 121		
Nom de la méthode	**ID de la méthode**	**Description de la méthode**
Null	0	Réservé par la présente norme pour usage futur
Reset	1	Réinitialisation des données d'application
Réservé pour usage futur par la présente norme	2..127	Ces identificateurs de méthode sont réservés pour usage futur par la présente norme
Implementation-specific use	128..255	Ces identificateurs de méthode sont disponibles pour une utilisation spécifique à une mise en œuvre

12.19.4.2 Méthode Reset

Le Tableau 285 spécifie les primitives de la méthode Reset.

Tableau 285 – Méthode Reset

<table>
<tr><th colspan="5">Nom du type d'objet normalisé: Application sublayer management object (ASLMO, objet de gestion de sous-couche d'application)</th></tr>
<tr><th colspan="5">Identificateur du type d'objet normalisé: 121</th></tr>
<tr><th>Nom de la méthode</th><th>ID de la méthode</th><th colspan="3">Description de la méthode</th></tr>
<tr><td rowspan="8">Reset</td><td>1</td><td colspan="3">Réinitialisation de sous-couche d'application</td></tr>
<tr><td colspan="4">Arguments d'entrée</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td>1</td><td>ResetType</td><td>Type: Unsigned8

Valeurs nommées:
0: non utilisé
1: réinitialiser aux valeurs de réglage d'usine par défaut
2: réinitialiser aux valeurs de réglage configurées
3: réinitialiser à chaud (réinitialisation aux valeurs de réglage configurées et toutes les éventuelles informations relatives au contrat de communication)
4: réinitialiser toutes les données dynamiques (relatives aux statistiques, par exemple)
Valeurs 5..255 réservées</td><td>Type de réinitialisation souhaité</td></tr>
<tr><td colspan="4">Arguments de sortie</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td colspan="4">Aucun</td></tr>
</table>

12.19.4.3 Arguments d'entrée

Le paramètre ResetType indique le type de réinitialisation souhaité. La sous-couche peut être réinitialisée:

- aux valeurs de réglage d'usine par défaut;
- pour seulement maintenir les valeurs de réglage configurées (le cas échéant);
- pour seulement maintenir le jeu de valeurs de réglage configurées et de contrat de communication (le cas échéant); ou
- pour toutes les statistiques dynamiques seulement.

NOTE Un exemple de valeurs de réglage d'usine par défaut est:

– paramètre de configuration MalformedAPDUsAdvise indiquant "disabled";

– paramètre de configuration TimeIntervalForCountingMalformedAPDUs indiquant 100 APDU; et

– paramètre de configuration MalformedAPDUsThreshold indiquant un intervalle de temps de zéro.

12.19.4.4 Arguments de sortie

Il n'y a aucun argument de sortie pour cette méthode.

12.19.4.5 Codes de réponse

Les codes suivants de retour d'informations sont valides pour cette méthode:

- success;
- invalidArgument; et
- ceux qui sont définis par le fournisseur.

12.19.5 Alertes de l'objet de gestion de sous-couche d'application

Le Tableau 286 définit les alertes pour l'ASLMO.

Tableau 286 – Alertes d'ASLMO

Nom(s) du type d'objet normalisé: Application sublayer management object (ASLMO, objet de gestion de sous-couche d'application)					
Identificateur du type d'objet normalisé: 121					
Description de l'alerte: Alerte APDU mal formée					
Classe d'alertes (Enumerated: alarme ou événement)	**Catégorie d'alertes (Enumerated: diagnostic d'appareil, diagnostic de comm., sécurité ou processus)**	**Type d'alerte (Enumerated: en fonction de la catégorie d'alertes)**	**Priorité d'alerte**	**Données associées**	**Description des données associées incluses à l'alerte**
Evénement	Communication diagnostic	malformedAPDU CommunicationAlert	7 (à savoir une alerte de priorité "medium" de mi-plage)	Aucun type normalisé spécial n'est défini, car le contenu de protocole ne correspond pas à un attribut de l'ASLMO. Plutôt, il est construit au sein du protocole comme une séquence implicite, dont la construction est identifiable par la combinaison de la classe d'alertes, de la catégorie d'alerte et du type d'alerte	Trois éléments sont inclus dans la séquence suivante: a) adresse source des APDU mal formées (adresse IPv6Address) b) valeur-seuil dépassée (Unsigned16) c) intervalle de temps dans lequel le seuil a été dépassé (TAITimeDifference)

12.19.6 Services de DMAP invoqués par la sous-couche d'application

Si l'intervalle des APDU mal formées de l'ASL et le seuil des APDU mal formées de l'ASL qui étaient configurés sont tous deux différents de zéro, l'ASL doit commencer à garder les statistiques des APDU mal formées. Si le seuil est atteint avant ou à l'expiration de l'intervalle de temps configuré, l'ASL doit rapporter au DMAP qu'il convient de générer une alerte de diagnostic de communication. Les données fournies par l'ASL au DMAP doivent inclure:

a) une indication que la situation malformedAPDUsThresholdReached a été atteinte; et

NOTE Cela est indiqué au cas où le seuil d'APDU mal formées serait atteint pour des APDU mal formées reçues en provenance d'une seule adresse N source dans l'intervalle configuré d'APDU mal formées d'ASL.

b) les informations de diagnostics relatives aux APDU mal formées détectées, telles que:
   - le nombre des APDU reçues qui n'avaient pas la longueur correcte;
   - le nombre des APDU reçues avec un identificateur de service non valide; et

– le nombre des APDU reçues avec un identificateur de service mal utilisé; et

c) l'adresse IPv6Address source des APDU mal formées; et

d) la valeur-seuil qui avait été atteinte; et

e) la durée temporelle sur laquelle les APDU mal formées avaient été reçues.

Cet intervalle de temps est calculé comme étant le temps allant de la détection de la première APDU issue de l'appareil indiqué par l'ASL jusqu'à la détection de l'APDU mal formée qui a conduit à ce que la limite du seuil d'ASL soit atteinte pour l'appareil indiqué. Cette durée temporelle doit être inférieure ou égale à l'intervalle de temps configuré qui est établi dans les paramètres de gestion d'ASL.

12.19.7 Objets normalisés des industries de transformation

12.19.7.1 Généralités

Les objets normalisés définis dans la présente norme sont inclus pour traiter des besoins fondamentaux de l'industrie de transformation. La théorie de terrain unifiée (pour le contrôle de processus) qui sous-tend la présente norme définit les objets de terrain normalisés en effectuant un effet de levier sur les définitions existantes d'objets d'appareils de terrain à partir de normes de contrôle de processus orientées objet et éprouvées sur le terrain. L'ensemble d'objets a été sélectionné selon des caractéristiques d'utilisation communes et il est défini pour permettre l'interfonctionnabilité et l'interopérabilité (dans son domaine d'application) limitée parmi des appareils.

NOTE 1 La terminologie utilisée dans le présent document, notamment SOE, PV, OP, OOS, MAN et AUTO, est la terminologie commune aux industries de transformation.

NOTE 2 La présente norme présume que des restrictions d'accès aux attributs d'objet, au besoin pour satisfaire à des exigences d'utilisation de système, sont mises en vigueur par des appareils d'interface humaine et/ou des appareils passerelles.

Pour prendre en charge les marqueurs temporels utilisés dans les rapports d'alarme dans l'industrie de contrôle de processus, la construction de valeur de temps utilisée pour représenter le temps doit prendre en charge une exactitude de codage de 1 ms. Cette exactitude est nécessaire pour prendre en charge une résolution à grande vitesse typique des applications de séquences d'événements (SOE) des industries de transformation.

12.19.7.2 Objets d'application utilisateur dans les industries de transformation

Une liste de base d'objets d'application utilisateur est anticipée pour le profil des industries de contrôle de processus. Les objets de terrain unifiés (UFO) définis dans la présente norme sont:

- analog input object (objet d'entrée analogique);
- analog output object (objet de sortie analogique);
- binary input object (objet d'entrée binaire);et
- binary output object(objet de sortie binaire).

Le mode de commande d'objet d'application prend en charge les modes suivants:

- Le mode cible est le mode auquel l'appareil a été commandé de passer. Il peut être différent du mode réel si l'appareil ne peut pas accepter le mode cible en raison d'une erreur, etc.
- Le mode réel est le mode courant de l'objet.
- Le mode normal est le mode de fonctionnement du bloc qui est désiré par le technicien de commande responsable. Le mode normal est l'un des modes autres qu'OOS, qui est spécifié comme étant le "fonctionnement normal" pour le bloc.
- Les modes autorisés représentent l'ensemble de modes qui sont valides pour cet objet. Il s'agit d'un filtre qui peut être appliqué pour limiter le mode cible du bloc. Par

exemple, le mode manuel peut être désactivé de cette façon. OOS est toujours inclus dans l'ensemble de modes autorisés.

Les modes suivants sont pris en charge:

– hors service (OOS): l'appareil ne mesure pas activement la valeur PV ou n'accepte pas la valeur OP. Un autre nom commun pour ce mode est "inactive". La valeur et le statut associé indiquant l'état d'OOS sont encore communiqués par l'appareil. Cela ne vise pas à désactiver la communication;
– manuel (MAN): la valeur de processus peut être saisie manuellement par l'opérateur, utilisée pour la commande de boucle ouverte avec humain dans la boucle ou pour neutraliser une mesure fautive. Egalement utile pour des essais; et
– automatique (AUTO): l'appareil mesure activement la valeur PV ou accepte la valeur OP.

Un attribut structuré doit être ajouté pour chaque type de rapport d'alerte pris en charge par l'objet. Cet attribut prend en charge l'activation/désactivation d'un rapport d'alerte et établit les priorités d'alertes.

Pour des alarmes, un attribut structuré peut en plus être exigé pour établir des limites d'alarme, pour indiquer si une alarme est présente.

12.19.7.3 Analog input user object (objet utilisateur d'entrée analogique)

12.19.7.3.1 Généralités

Un objet utilisateur normalisé d'entrée analogique représentant une encapsulation d'une entrée analogique est défini. Si plusieurs entrées analogiques sont représentées par un appareil, il convient que plusieurs objets utilisateur d'entrée analogiques soient instanciés. Les attributs spécifiques à un type d'objet relatifs à cet objet incluent:

- process value: une valeur en virgule flottante représentée en unités d'étude et statut;
- mode: un attribut structuré représentant un mode cible, un mode réel, un mode autorisé, et un mode normal;
- corresponding concentrator object: spécifie le concentrateur associé pour éditer la PV; et
- scale: représente la plage et les unités de la valeur de processus par l'intermédiaire d'un attribut structuré qui indique les limites 0 % et 100 % de la plage de valeurs nominale, une représentation codée d'unités d'étude, ainsi que le nombre de chiffres significatifs qu'il convient d'utiliser pour l'affichage

Des alertes normalisées pour cet objet seront également définies.

12.19.7.3.2 Attributs d'objet

Un objet d'entrée analogique a les attributs définis dans le Tableau 287.

Tableau 287 – Attributs de l'objet d'entrée analogique

Nom du type d'objet normalisé: Analog input object (objet d'entrée analogique)				
Identificateur du type d'objet normalisé: 99				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
ObjectIdentifier	Identificateur-clé d'objet	Identificateur unique pour l'objet	Type: Unsigned16 Classification: Constant Plage valide: > 0	N/A
Réservée pour un usage futur	0	-	-	-
PV	1	Variable de mesure en unités d'étude du capteur	Type: Process control value and status for analog value Classification: Dynamic Accessibilité: Lecture seule Valeur par défaut: NaN Etat: Inconnu; Sous-état: Inconnu; Condition limite: Non limité Plage valide: Voir la définition de la structure process control value and status for analog value	Valeur de processus analogique et statut pour la valeur en question. L'accessibilité s'effectue uniquement en lecture/écriture lorsque MODE.Target=MAN. Voir 12.19.7.2. Lorsqu'une écriture dans la PV est effectuée, l'appareil peut mettre cela en œuvre comme une écriture dans une variable interne non visible du réseau et utiliser la valeur non visible pour construire la valeur qu'elle représente pour la PV. Selon le cas, l'appareil peut rapporter pour la PV un statut différent de celui qui avait été fourni dans la demande d'écriture.
Mode	2	Mode	Type: Process control mode Classification: Static Accessibilité: Lecture seule pour le mode réel; le mode cible, le mode autorisé et le mode normal ont tous un accès en lecture/écriture Valeur par défaut: La valeur du mode réel indique OOS Plage valide: Voir la définition du type de structure Process control mode	Valeurs de mode réel, de mode cible, de mode autorisé et de mode normal
Réservée pour un usage futur	3	-	-	-

Nom du type d'objet normalisé: Analog input object (objet d'entrée analogique)				
Identificateur du type d'objet normalisé: 99				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Echelle	4	Plage et unités de la variable de mesure	Type: Process control scaling data	Information de mise à l'échelle pour la valeur de processus analogique
			Classification: Static	
			Accessibilité: Lecture/écriture	
			Valeur par défaut: Les valeurs des unités d'étude pour 0 % et pour 100 % indiquent TOUTES LES DEUX 0	
			Plage valide: Voir la définition du type de structure scale	
Réservé pour usage futur par la présente norme	5..25	-	-	*N* octets de contenu présentement indéfini

12.19.7.3.3 Méthodes d'objets normalisés

Un objet d'entrée analogique a les méthodes définies dans le Tableau 288.

Tableau 288 – Méthodes de l'objet d'entrée analogique

Nom du type d'objet normalisé: Analog input object (objet d'entrée analogique)		
Identificateur du type d'objet normalisé: 99		
Nom de la méthode	**ID de la méthode**	**Description de la méthode**
Null	0	Réservé par la présente norme pour usage futur
Réservé pour usage futur par la présente norme	0..127	Ces identificateurs de méthode sont réservés pour usage futur par la présente norme
Implementation-specific use	128..255	Ces identificateurs de méthode sont disponibles pour une utilisation spécifique à une mise en œuvre

12.19.7.3.4 Alertes

Une entrée analogique peut rapporter les alertes montrées dans le Tableau 289. Si une alerte est prise en charge, un attribut de descripteur d'alerte correspondant doit être ajouté à l'objet d'entrée analogique pour décrire les caractéristiques de l'alerte.

Tableau 289 – Alertes d'entrée analogique

Nom(s) du type d'objet normalisé: Analog input						
Identificateur du type d'objet normalisé: 99						
Description de l'alerte: alertes d'entrée analogique						
Classe d'alertes (Enumerated: alarme ou événement)	**Catégorie d'alertes (Enumerated: diagnostic d'appareil, diagnostic de comm., sécurité ou processus)**	**Type(s) d'alerte (Enumerated: en fonction de la catégorie d'alertes)**		**Priorité d'alerte**	**Données associées: type et taille**	**Description des données associées incluses à l'alerte**
Alarme	Processus	1	Elevée	Toute	Type: Float32	Variable de processus
Alarme	Processus	2	HighHigh	Toute	Type: Float32	Variable de processus
Alarme	Processus	3	Faible	Toute	Type: Float32	Variable de processus
Alarme	Processus	4	LowLow	Toute	Type: Float32	Variable de processus
Alarme	Processus	5	DeviationLow	Toute	Type: Float32	Variable de processus
Alarme	Processus	6	DeviationHigh	Toute	Type: Float32	Variable de processus
Alarme	Processus	0	OutOfService	Toute	Type: Float32	Variable de processus

12.19.7.4 Analog output user object (objet utilisateur de sortie analogique)

12.19.7.4.1 Généralités

Un objet normalisé utilisateur de sortie analogique représente une encapsulation d'une sortie analogique. Si plusieurs sorties analogiques sont représentées par un appareil, il convient que plusieurs objets utilisateur de sorties analogiques soient instanciés. Les attributs spécifiques à un type d'objet relatifs à cet objet incluent:

- commanded output value: une valeur en virgule flottante représentée en unités d'étude et statut;
- mode: un attribut structuré représentant un mode cible, un mode réel, un mode autorisé, et un mode normal;
- Readback: valeur et statut de la position réelle;
- provider of OP value: indique la source de la valeur OP;
- corresponding concentrator object: spécifie le concentrateur associé pour éditer la valeur Readback; et
- scale: représente la plage et les unités de la valeur de processus par l'intermédiaire d'un attribut structuré qui indique les limites 0 % et 100 % de la plage de valeurs nominale, une représentation codée d'unités d'étude, ainsi que le nombre de chiffres significatifs qu'il convient d'utiliser pour l'affichage

12.19.7.4.2 Attributs d'objet

Un objet de sortie analogique a les attributs définis dans le Tableau 290.

Tableau 290 – Attributs de sortie analogique *(1 de 2)*

Nom du type d'objet normalisé: Analog output object (objet de sortie analogique)				
Identificateur du type d'objet normalisé: 98				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
ObjectIdentifier	Identificateur-clé d'objet	Identificateur unique pour l'objet	Type: Unsigned16 Classification: Constant Plage valide: > 0	N/A
Réservée pour un usage futur	0	-	-	-
OP	1	Valeur de sortie pour l'actionneur	Type: Process control value and status for analog value structure Classification: Dynamic Accessibilité: Lecture/écriture Valeur par défaut: NaN Etat: Inconnu; Sous-état: Inconnu; Condition limite: Non limité Plage valide: Voir la définition de la structure process control value and status for analog value	Il s'agit de l'attribut normalisé qui sert d'attribut de destination pour une publication issue d'un autre objet
Mode	2	Mode	Type: Process control mode Classification: Static Accessibilité: Lecture seule pour le mode réel; le mode cible, le mode autorisé et le mode normal ont tous un accès en lecture/écriture Valeur par défaut: La valeur du mode réel indique OOS Plage valide: Voir la définition du type de structure Process control mode	Valeurs de mode réel, de mode cible, de mode autorisé et de mode normal
Réservée pour un usage futur	3	-	-	-
Readback	4	Valeur Readback – la position réelle de l'OP	Type: Process control value and status for analog value Classification: Dynamic Accessibilité: Lecture seule Valeur par défaut: NaN; Etat: Inconnu; Sous-état: Inconnu; Condition limite: Non limité Plage valide: Voir la définition de la structure process control value and status for analog value	Valeur de processus analogique et statut pour la valeur en question. Il s'agit de l'attribut normalisé qui est édité à partir de cet objet. Si cet objet est étendu et, de ce fait, un autre attribut a besoin d'être édité, un/des objet(s) concentrateur(s) représente(nt) la/les publication(s) résultante(s)

Tableau 290 *(2 de 2)*

Nom du type d'objet normalisé: Analog output object (objet de sortie analogique)				
Identificateur du type d'objet normalisé: 98				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Réservée pour un usage futur	5	-	-	-
Echelle	6	Plage et unités de la variable de sortie	Type: Process control scaling data structure	Informations de mise à l'échelle
			Classification: Static	
			Accessibilité: Lecture/écriture	
			Valeur par défaut: Les valeurs des unités d'étude pour 0 % et pour 100 % indiquent TOUTES LES DEUX 0	
			Plage valide: Voir la définition du type de structure Scale	
Réservé pour usage futur par la présente norme	7..25	-	-	*N* octets de contenu présentement indéfini

12.19.7.4.3 Méthodes d'objets normalisés

Un objet de sortie analogique a les méthodes définies dans le Tableau 291.

Tableau 291 – Méthodes de l'objet de sortie analogique

Nom du type d'objet normalisé: Analog output object (objet de sortie analogique)		
Identificateur du type d'objet normalisé: 98		
Nom de la méthode	**ID de la méthode**	**Description de la méthode**
Null	0	Réservé par la présente norme pour usage futur
Réservé pour usage futur par la présente norme	0..127	Ces identificateurs de méthode sont réservés pour usage futur par la présente norme.
Implementation-specific use	128..255	Ces identificateurs de méthode sont disponibles pour une utilisation spécifique à une mise en œuvre

12.19.7.4.4 Alertes

Une sortie analogique peut rapporter les alertes montrées dans le Tableau 292. Si une alerte est prise en charge, un attribut de descripteur d'alerte correspondant doit être ajouté à l'objet de sortie analogique pour décrire les caractéristiques de l'alerte.

Tableau 292 – Alertes de sortie analogique

Nom(s) du type d'objet normalisé: Sortie analogique						
Identificateur du type d'objet normalisé: 98						
Description de l'alerte: objet de sortie analogique						
Classe d'alertes (Enumerated: alarme ou événement)	**Catégorie d'alertes (Enumerated: diagnostic d'appareil, diagnostic de comm., sécurité ou processus)**	**Type(s) d'alerte (Enumerated: en fonction de la catégorie d'alertes)**		**Priorité d'alerte**	**Données associées: type et taille**	**Description des données associées incluses à l'alerte**
Alarme	Processus	1	Elevée	Toute	Type: Float32	Variable de processus
Alarme	Processus	2	HighHigh	Toute	Type: Float32	Variable de processus
Alarme	Processus	3	Faible	Toute	Type: Float32	Variable de processus
Alarme	Processus	4	LowLow	Toute	Type: Float32	Variable de processus
Alarme	Processus	5	DeviationLow	Toute	Type: Float32	Variable de processus
Alarme	Processus	6	DeviationHigh	Toute	Type: Float32	Variable de processus
Alarme	Processus	0	OutOfService	Toute	Type: Float32	Variable de processus

12.19.7.5 Binary input user object (objet utilisateur d'entrée binaire)

12.19.7.5.1 Généralités

Un objet normalisé utilisateur d'entrée binaire représente une encapsulation d'une seule entrée binaire. Si plusieurs entrées binaires sont représentées par un appareil, il convient que plusieurs objets utilisateur d'entrée binaires existent pour les représenter. Les attributs spécifiques à un type d'objet relatifs à cet objet incluent:

- process value: valeur binaire et statut;
- mode: un attribut structuré représentant un mode cible, un mode réel, un mode autorisé, et un mode normal; et
- corresponding concentrator object: spécifie le concentrateur associé pour éditer la valeur de processus.

12.19.7.5.2 Attributs d'objet

Un objet d'entrée binaire a les attributs définis dans le Tableau 293.

Tableau 293 – Attributs de l'objet d'entrée binaire

Nom du type d'objet normalisé: Binary input object (objet d'entrée binaire)				
Identificateur du type d'objet normalisé: 97				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
ObjectIdentifier	Identificateur-clé d'objet	Identificateur unique pour l'objet	Type: Unsigned16 Classification: Constant Plage valide: > 0	N/A
Réservée pour un usage futur	0	-	-	-
PV_B	1	Variable de mesure discrète	Type: Process control value and status for discrete value Classification: Dynamic Accessibilité: Lecture seule Valeur par défaut: 0 Etat: Inconnu; Sous-état: Inconnu; Condition limite: Non limité Plage valide: Voir la définition de la structure process control value and status for discrete value	Valeur de processus binaire et statut pour la valeur en question. Il s'agit de l'attribut normalisé qui est édité à partir de cet objet. Si cet objet est étendu et, de ce fait, un autre attribut a besoin d'être édité, un/des objet(s) concentrateur(s) représente(nt) la/les publication(s) résultante(s). L'accessibilité s'effectue uniquement en lecture/écriture dans le mode MAN. Voir 12.19.7.2. Lorsqu'une écriture dans la PV_B est effectuée, l'appareil peut mettre cela en œuvre comme une écriture dans une variable interne non visible du réseau et utiliser la valeur non visible pour construire la valeur qu'elle représente pour la PV_B. Selon le cas, l'appareil peut rapporter pour la PV_B un statut différent de celui qui avait été fourni dans la demande d'écriture.
Mode	2	Mode	Type: Process control mode Classification: Static Accessibilité: Lecture seule pour le mode réel; le mode cible, le mode autorisé et le mode normal ont tous un accès en lecture/écriture Valeur par défaut: La valeur du mode réel indique OOS Plage valide: Voir la définition du type de structure Process control mode	Valeurs de mode réel, de mode cible, de mode autorisé et de mode normal
Réservé pour usage futur par la présente norme	3..25	-	-	*N* octets de contenu présentement indéfini

12.19.7.5.3 Méthodes d'objets normalisés

Un objet d'entrée binaire a les méthodes définies dans le Tableau 294.

Tableau 294 – Méthodes de l'objet d'entrée binaire

Nom du type d'objet normalisé: Binary input object (objet d'entrée binaire)		
Identificateur du type d'objet normalisé: 97		
Nom de la méthode	**ID de la méthode**	**Description de la méthode**
Null	0	Réservé par la présente norme pour usage futur
Réservé pour usage futur par la présente norme	0..127	Ces identificateurs de méthode sont réservés pour usage futur par la présente norme
Spécifique à une mise en œuvre	128..255	Ces identificateurs de méthode sont disponibles pour une utilisation spécifique à une mise en œuvre

12.19.7.5.4 Alertes

Une sortie analogique peut rapporter les alertes montrées dans le Tableau 295. Si une alerte est prise en charge, un attribut de descripteur d'alerte correspondant doit être ajouté à l'objet d'entrée binaire pour décrire les caractéristiques de l'alerte.

Tableau 295 – Alertes d'entrée binaire

Nom(s) du type d'objet normalisé: Binary input object (objet d'entrée binaire)						
Identificateur du type d'objet normalisé: 97						
Description de l'alerte: alertes d'entrée binaire						
Classe d'alertes (Enumerated: alarme ou événement)	**Catégorie d'alertes (Enumerated: diagnostic d'appareil, diagnostic de comm., sécurité ou processus)**	**Type(s) d'alerte (Enumerated: en fonction de la catégorie d'alertes)**		**Priorité d'alerte**	**Données associées: type et taille**	**Description des données associées incluses à l'alerte**
Alarme	Processus	1	DiscreteAlarm	Toute	Type: Boolean	Process value
Alarme	Processus	0	OutOfService	Toute	Type: Boolean	Process value

12.19.7.6 Binary output user object (objet utilisateur de sortie binaire)

12.19.7.6.1 Généralités

Un objet normalisé utilisateur de sortie binaire représente une encapsulation d'une seule sortie binaire. Si plusieurs sorties binaires sont représentées par un appareil, il convient que plusieurs objets utilisateur de sorties binaires existent pour les représenter. Les attributs spécifiques à un type d'objet relatifs à cet objet incluent:

- commanded output value: valeur binaire et statut;
- mode: un attribut structuré représentant un mode cible, un mode réel, un mode autorisé, et un mode normal;
- provider of OP value: indique la source de la valeur OP;
- Readback value: valeur binaire et statut; et
- corresponding concentrator object: spécifie le concentrateur associé pour éditer la valeur Readback_B.

12.19.7.6.2 Attributs d'objet

Un objet de sortie binaire a les attributs définis dans le Tableau 296.

Tableau 296 – Attributs de sortie binaire

Nom du type d'objet normalisé: Binary output object (objet de sortie binaire)				
Identificateur du type d'objet normalisé: 96				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
ObjectIdentifier	Identificateur-clé d'objet	Identificateur unique pour l'objet	Type: Unsigned16 Classification: Constant Plage valide: > 0	N/A
Réservée pour un usage futur	0	-	-	-
OP_B	1	Variable de mesure discrète	Type: Process control value and status for discrete value structure Classification: Dynamic Accessibilité: Lecture/écriture Valeur par défaut: 0 Etat: Inconnu; Sous-état: Inconnu; Condition limite: Non limité Plage valide: Voir la définition de la structure process control value and status for discrete value	Il s'agit de l'attribut normalisé qui est l'attribut de destination pour une publication issue d'un autre objet
Mode	2	Mode	Type: Process control mode Classification: Static Accessibilité: Lecture seule pour le mode réel; le mode cible, le mode autorisé et le mode normal ont tous un accès en lecture/écriture Valeur par défaut: La valeur du mode réel indique OOS Plage valide: Voir la définition du type de structure Process control mode	Valeurs de mode réel, de mode cible, de mode autorisé et de mode normal
Réservée pour un usage futur	3	-	-	-
Readback_B	4	Valeur Readback – la position réelle de l'actionneur	Type: Process control value and status for discrete value Classification: Dynamic Accessibilité: Lecture seule Valeur par défaut: 0; Etat: Inconnu; Sous-état: Inconnu; Condition limite: Non limité Plage valide: Voir la définition de la structure process control value and status for analog value	Valeur de processus analogique et statut pour la valeur en question. Il s'agit de l'attribut normalisé qui est édité à partir de cet objet. Si cet objet est étendu et, de ce fait, un autre attribut a besoin d'être édité, un/des objet(s) concentrateur(s) représente(nt) la/les publication(s) résultante(s).
Réservé pour usage futur par la présente norme	5..25	-	-	*N* octets de contenu présentement indéfini

12.19.7.6.3 Méthodes d'objets normalisés

Un objet de sortie binaire a les méthodes définies dans le Tableau 297.

Tableau 297 – Méthodes de l'objet de sortie binaire

Nom du type d'objet normalisé: Binary output object (objet de sortie binaire)		
Identificateur du type d'objet normalisé: 96		
Nom de la méthode	**ID de la méthode**	**Description de la méthode**
Null	0	Réservé par la présente norme pour usage futur
Réservé pour usage futur par la présente norme	0..127	Ces identificateurs de méthode sont réservés pour usage futur par la présente norme
Implementation-specific use	128..255	Ces identificateurs de méthode sont disponibles pour une utilisation spécifique à une mise en œuvre

12.19.7.6.4 Alertes

Une sortie binaire peut rapporter les alertes montrées dans le Tableau 298. Si une alerte est prise en charge, un attribut de descripteur d'alerte correspondant doit être ajouté à l'objet de sortie binaire pour décrire les caractéristiques de l'alerte.

Tableau 298 – Alertes de sortie binaire

Nom(s) du type d'objet normalisé: Binary output object (objet de sortie binaire)						
Identificateur du type d'objet normalisé: 96						
Description de l'alerte: alertes de sortie binaire						
Classe d'alertes (Enumerated: alarme ou événement)	**Catégorie d'alertes (Enumerated: diagnostic d'appareil, diagnostic de comm., sécurité ou processus)**	**Type(s) d'alerte (Enumerated: en fonction de la catégorie d'alertes)**		**Priorité d'alerte**	**Données associées: type et taille**	**Description des données associées incluses à l'alerte**
Alarme	Processus	1	DiscreteAlarm	Toute	Type: Boolean	Process value
Alarme	Processus	0	OutOfService	Toute	Type: Boolean	Process value

12.19.8 Profil des industries d'automation d'usine

12.19.8.1 Généralités

Des objets normalisés complémentaires pour prendre en charge les besoins de l'automation d'usine peuvent être ajoutés dans de futures versions de la présente norme. Une idée plus détaillée, spécifique à l'automation d'usine, est accordée à une version ultérieure de l'activité de normalisation de la présente norme. Les exemples des objets qui peuvent être définis pour satisfaire aux besoins de l'automation d'usine peuvent inclure:

a) binary input user object (un contact, par exemple);

b) binary output user object (une bobine, par exemple);

c) analog input user object;

d) analog output user object;

e) register input user object (par exemple, pour établir un mapping de 16 bits d'informations à deux états souvent trouvées dans les registres d'entrée de PLC);

f) register output user object, (par exemple, pour établir un mapping de 16 bits d'informations à deux états souvent trouvées dans les registres de sortie de PLC);

g) multi-state input user object (par exemple, pour établir un mapping de 8 bits d'informations d'état pour une soupape avec des états énumérés tels que s'ouvrant, ouverte, se fermant, fermée ou pour établir un mapping d'un moteur unidirectionnel avec des états tels que éteint, démarrant, en marche, s'arrêtant); et

h) multi-state output user object (par exemple, pour établir un mapping de 8 bits d'informations de sortie d'état pour un appareil de sortie avec des états énumérés).

Des alertes normalisées pour ces objets peuvent aussi être définies.

12.19.8.2 Objets spécifiques à un fabricant

Les fournisseurs peuvent aussi définir des objets personnalisés spécifiques à un fournisseur ou spécifiques à un appareil. Un exemple d'un objet spécifique à un fournisseur pour les systèmes de processus est un objet d'affichage équipé dans lequel une chaîne peut être stockée pour être montrée à une personne à proximité de l'appareil.

12.20 Structures de données normalisées de l'industrie de contrôle de processus

12.20.1 Généralités

Les structures de données normalisées utilisées dans l'industrie de contrôle de processus doivent être les structures de données acheminées par le protocole défini par la présente norme. Les structures de données normalisées indépendantes vis-à-vis de toute industrie et les structures de données dans l'industrie de contrôle de processus sont toutes les deux résumées en Annexe L.

NOTE Les définitions de structures de données spécifiques à un fournisseur ne sont pas prises en charge.

12.20.2 Statut pour des informations analogiques

Le statut pour des informations analogiques est un octet de format fixe condensé contenant les éléments montrés dans le Tableau 299.

Tableau 299 – Octet Status

<table>
<tr><th colspan="2">Bit 7 (MSB)</th><th>Bit 6</th><th>Bit 5</th><th>Bit 4</th><th>Bit 3</th><th>Bit 2</th><th>Bit 1</th><th>Bit 0 (LSB)</th></tr>
<tr><td colspan="2">Qualité</td><td></td><td><réservé></td><td colspan="3">Sous-état dépendant de la qualité</td><td colspan="2">Condition limite</td></tr>
<tr><td colspan="2">0 = bad
(il convient de ne pas utiliser la valeur)</td><td></td><td rowspan="4">Ce bit doit toujours être mis à zéro</td><td colspan="3">Valeurs nommées lorsque la qualité=bad:
0: non spécifique;
1: erreur de configuration;
2: non connecté;
3: défaillance d'appareil;
4: défaillance de capteur;
5: pas de communication avec lastUsableValue;
6: pas de communication sans lastUsableValue;
7: hors service (la valeur n'est pas calculée)</td><td colspan="2" rowspan="4">Valeurs nommées:
0: non limité;
1: limite basse;
2: limite haute;
3: constant (à la fois limite haute et limite basse)</td></tr>
<tr><td colspan="2">1 = uncertain
(valeur d'une qualité inférieure à la normale)</td><td></td><td colspan="3">Valeurs nommées lorsque la qualité=uncertain:
0: non spécifique;
1: dernière valeur utilisable;
2: entrée substituée ou manuelle;
3: valeur initiale;
4: conversion de capteur inexacte;
5: limites de plage dépassées;
6: en dessous de la normale;
7: réservée</td></tr>
<tr><td colspan="2">2 = good
(la qualité de la valeur est bonne, mais une condition d'alarme peut exister)</td><td></td><td colspan="3">Valeurs nommées lorsque la qualité=good:
0: il n'existe pas de condition spéciale;
1..7: réservées</td></tr>
<tr><td colspan="2">3 = reserved</td><td></td><td colspan="3">Toutes les valeurs sont réservées.
Au sein de la présente norme, ils doivent toujours être mis à zéro</td></tr>
<tr><td colspan="9">NOTE Les définitions pour l'octet de statut sont un sous-ensemble de celles définies par la HART Communication Foundation, la Fieldbus Foundation, et l'OPC Foundation.</td></tr>
</table>

12.20.3 Valeur et statut pour les informations analogiques

Sachant que le statut n'indique pas une substitution, les écritures déclenchées par un réseau en utilisant ces structures de types de données analogiques ne sont pas autorisées par la présente norme.

La structure des informations analogiques est montrée dans le Tableau 300.

Tableau 300 – Type de données: Valeur de contrôle de processus et statut pour la valeur analogique

Nom du type de données normalisé: Valeur de contrôle de processus et statut pour la valeur analogique		
Code du type de données normalisé: 65		
Nom de l'élément	**Identificateur de l'élément**	**Type scalaire de l'élément**
Status	1	Type: BitString8 Classification: Dynamic Ensemble de valeurs valides: Voir Tableau 299
Value	2	Type: Float32 Classification: Dynamic

12.20.4 Valeur et statut pour les informations binaires

Sachant que le statut n'indique pas une substitution, les écritures déclenchées par un réseau dans ces structures de types de données numériques ne sont pas autorisées.

La structure des informations numériques est montrée dans le Tableau 301.

Tableau 301 – Type de données: Valeur de contrôle de processus et statut pour la valeur binaire

Nom du type de données normalisé: Valeur de contrôle de processus et statut pour la valeur binaire		
Code du type de données normalisé: 66		
Nom de l'élément	**Identificateur de l'élément**	**Type scalaire de l'élément**
Status	1	Type: BitString8 Accessibilité: Peut varier selon l'utilisation Ensemble de valeurs valides: Voir Tableau 299
Valeur	2	Type: Boolean

12.20.5 Mode contrôle de processus

Les éléments dans le "process control mode" sont montrés dans le Tableau 302.

Tableau 302 – Type de données: Mode contrôle de processus

Nom du type de données normalisé: Mode contrôle de processus		
Code du type de données normalisé: 69		
Nom de l'élément	**Identificateur de l'élément**	**Type scalaire de l'élément**
Cible	1	Type: BitString8 Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: OOS Ensemble de valeurs valides: Voir Tableau 303
Actual	2	Type: BitString8 Classification: Dynamic Accessibilité: Lecture seule Valeur par défaut: OOS Ensemble de valeurs valides: Voir Tableau 303
Autorisé	3	Type: BitString8 Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: OOS Ensemble de valeurs valides: Voir Tableau 303
Valeur normale	4	Type: BitString8 Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: OOS Ensemble de valeurs valides: Voir Tableau 303

La valeur de chaque élément de la structure mode est représentée par un bitstring contenant les bits dans le Tableau 303, ou les bits <réservé> doivent être mis à zéro (0).

Tableau 303 – Type de données: Bitstring de mode contrôle de processus

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<réservé>	<réservé>	<réservé>	AUTO	MAN	<réservé>	<réservé>	OOS

A savoir:

- OOS seul est égal à 0x01, avec la valeur décimale équivalente de 1.
- MAN seul est égal à 0x08, avec la valeur décimale équivalente de 8.
- AUTO seul est égal à 0x10, avec la valeur décimale équivalente de 16.

12.20.6 Mise à l'échelle

Les éléments dans le "process control scaling" (mise à l'échelle de contrôle de processus sont montrés dans le Tableau 304.

Tableau 304 – Type de données: Mise à l'échelle du contrôle de processus

Nom du type de données normalisé: Mise à l'échelle du contrôle de processus		
Code du type de données normalisé: 68		
Nom de l'élément	**Identificateur de l'élément**	**Type scalaire de l'élément**
Unités d'étude à 100 % (la plage supérieure du paramètre d'objet associé)	1	Type: Float32 Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: 0
Unités d'étude à 0 % (la plage inférieure du paramètre d'objet associé)	2	Type: Float32 Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: 0
Index des unités pour les unités SI et les unités traditionnelles (non SI) [a] – plage 1000..1999 – autres codes réservés	3	Type: Unsigned16 Classification: Static Accessibilité: Lecture/écriture
Emplacement de la virgule décimale / du signe décimal (représente le nombre de chiffres à la droite de la virgule décimale/du signe décimal, à savoir, la signification de la partie fractionnaire de la valeur associée)	4	Type: Unsigned8 Classification: Static Accessibilité: Lecture/écriture Valeur par défaut: 0
[a] Telles que spécifiées dans *Standard Units Codes Table*, édité dans *Handbook of Measurement Equations and Tables* de l'ISA, 2nde Edition, ISBN 978 1 55617 946 4. Cet index est également édité par la Fieldbus Foundation sous TN-016:2007, Table 3.1.		

12.21 Tables complémentaires

12.21.1 Objets normalisés de profils de contrôle de processus

Le Tableau 305 énumère les objets normalisés de profils de contrôle de processus.

Tableau 305 – Objets normalisés de contrôle de processus

Type d'objet	Défini par	Identificateur du type d'objet normalisé
Analog input	12.19.7.3	99
Analog output	12.19.7.4	98
Binary input	12.19.7.5	97
Binary output	12.19.7.6	96

12.21.2 Services

Le Tableau 306 donne une liste de services.

Tableau 306 – Services

Service	Utilisation
Read	Lire la valeur d'un ou plusieurs attributs dans un ou plusieurs objets d'un UAP
Write	Ecrire la valeur dans un ou plusieurs attributs d'un ou plusieurs objets d'un UAP
Execute	Exécuter un ensemble de fonctions sur des instances d'objets d'un UAP
Publish	Editer périodiquement des données vers les abonnés
AlertReport	Rapporter une ou plusieurs conditions d'alerte
AlertAcknowledge	Acquitter un AlertReport
Tunnel	Passer la charge utile

12.22 Codage

12.22.1 Généralités

Les conditions pour coder des APDU sans fil dans la présente norme incluent les considérations suivantes:

- Certains messages se produisent souvent, comme les publications périodiques.
- Il convient que les messages soient courts, pour préserver l'énergie des batteries.
- Il convient d'avoir une sélection minimale de types de services d'ASL.

La taille maximale d'une APDU (qui est une TSDU) est déterminée en soustrayant (la taille des informations de TL s'ajoute à la TSDU pour former la TPDU) de (Assigned_Max_NSDU_Size), où Assigned_Max_NSDU_Size est un paramètre de sortie de la méthode utilisée pour établir un contrat de communication. A savoir:

maxAPDUSize = Assigned_Max_NSDU_Size – sizeOF(TLInfoAddedtoAPDU)

Voir 6.3.11.2.5 pour plus de détails relatifs à l'établissement de contrat de communication.

Le codage d'ASL doit assurer que la taille d'APDU maximale n'est pas dépassée.

NOTE L'IETF RFC 2348 fournit des recommandations relatives à la taille maximale des NPDU.

12.22.2 Règles de codage pour unités de données de protocole d'application

12.22.2.1 Généralités

Les règles de codage définies en 12.22.2 représentent le bit 0 comme étant le bit de poids faible (LSB) dans la valeur représentée.

Toutes les APDU contiennent un en-tête d'AL et un contenu d'APDU spécifique à un type de service. Le Tableau 307 indique la construction de codage générale d'une APDU.

Tableau 307 – Format de messagerie d'application

Peut être de 1 octet, 2 octets, 3 octets, ou 5 octets (voir 12.22.2.3)	***N* octets**
En-tête d'ASDU (assure le routage vers les objets corrects; fournit l'identification de type de service)	Contenu spécifique à un service

12.22.2.2 Concaténation

Les APDU peuvent être concaténées ensemble, et la concaténation d'APDU individuelles peut être donnée à la TL comme étant une seule TSDU. La taille de cette TSDU ne doit pas dépasser la taille d'APDU maximale pour des communications relativement au contrat de communication correspondant entre les applications expéditrice et réceptrice.

Le Tableau 308 décrit le format d'APDU concaténées en une seule TSDU.

Tableau 308 – APDU concaténées en une seule TSDU

APDU_1	**...**	**APDU_n**

La concaténation peut être utilisée pour s'assurer que lorsque l'une des APDU concaténées est reçue, toutes ont été reçues, ce qui fournit une base pour les actions multicomposants qui sont atomiques dans le cadre des communications inter-appareils.

NOTE La façon dont l'ASL détermine quand et quoi concaténer relève d'une initiative locale.

12.22.2.3 En-tête d'AL

L'en-tête d'AL prend en charge quatre modes d'adressage d'identificateur d'objet qui déterminent la construction d'en-tête au-delà du premier octet. Les modes d'adressage d'identificateur d'objet sont:

- mode d'adressage d'identificateur d'objet à quatre bits;
- mode d'adressage d'identificateur d'objet à huit bits (1 octet);
- mode d'adressage d'identificateur d'objet à seize bits (2 octets); et
- mode d'adressage inféré, qui peut être utilisé seulement dans la seconde APDU et les APDU suivantes d'une TSDU qui contient plusieurs APDU concaténées.

L'identification de l'UAP contenant l'objet est une fonction de la TL.

Le mode d'adressage d'identificateur d'objet en utilisation pour l'interprétation des APDU est indiqué dans des bits 5 et 6 du premier octet de l'en-tête d'APDU. Le Tableau 309 représente le codage de ce premier octet d'en-tête d'APDU.

Tableau 309 – Adressage d'objet

Octets	Bits							
	7	6	5	4	3	2	1	0
1	Type de primitive de service (.req = 0 .resp = 1)	Mode d'adressage d'identificateur d'objet Valeurs nommées: 00: mode à 4 bits 01: mode à 8 bits 10: mode à 16 bits 11: mode inféré		Type de service d'ASL				

12.22.2.4 Code d'identificateur d'objet

12.22.2.4.1 Généralités

Dans toutes les constructions d'en-tête, l'identificateur d'objet source représente l'identificateur d'objet dans l'application qui initie la primitive de service d'ASL (c'est-à-dire, l'initiateur d'une primitive .request ou d'une primitive .response), et l'identificateur d'objet de destination représente l'identificateur d'objet dans l'application qui reçoit la primitive de service d'ASL (c'est-à-dire, le destinataire d'une primitive .indication ou d'une primitive .confirmation).

12.22.2.4.2 Mode d'adressage d'identificateur d'objet à quatre bits

Le mode d'adressage d'identificateur d'objet à quatre bits indique que l'identificateur d'objet source et l'identificateur d'objet de destination peuvent être exprimés chacun sous la forme d'un nombre entier non signé de 4 bits. Ce mode d'adressage permet la prise en charge du compactage d'en-tête optimale pour les processus d'application avec un petit nombre d'objets. Ce mode est décrit en Tableau 310.

Tableau 310 – Construction d'en-tête d'APDU de mode d'adressage à quatre bits

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	Type de primitive de service	0	0	Type de service d'ASL				
1	Source object identifier				Destination object identifier			

12.22.2.4.3 Mode d'adressage d'identificateur d'objet à huit bits

Le mode d'adressage d'identificateur d'objet à huit bits indique que l'identificateur d'objet source et l'identificateur d'objet de destination peuvent être exprimés chacun sous la forme d'un nombre entier non signé de 8 bits, et en outre qu'au moins l'un des identificateurs d'objet ne peut pas être exprimé sous la forme d'un nombre entier non signé de 4 bits. Ce mode est décrit en Tableau 311.

Tableau 311 – Construction d'en-tête d'APDU de mode d'adressage à huit bits

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	Type de primitive de service	0	1	Type de service d'ASL				
1	Source object identifier							
1	Destination object identifier							

12.22.2.4.4 Mode d'adressage d'identificateur d'objet à seize bits

Le mode d'adressage d'identificateur d'objet à seize bits indique que l'identificateur d'objet source et l'identificateur d'objet de destination peuvent être exprimés chacun sous la forme d'un nombre entier non signé de 16 bits et en outre qu'au moins l'un des identificateurs d'objet ne peut pas être exprimé sous la forme d'un nombre entier non signé de 8 bits. Ce mode est décrit en Tableau 312.

Tableau 312 – Construction d'en-tête d'APDU de mode d'adressage à seize bits

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	Type de primitive de service	1	0	Type de service d'ASL				
2	Source object identifier							
2	Destination object identifier							

12.22.2.4.5 Mode d'adressage d'identificateur d'objet inféré pour des concaténations optimisées

Le mode d'adressage d'identificateur d'objet inféré est une technique d'optimisation utilisée seulement au sein d'une APDU concaténée. L'intention de cette technique est de sauvegarder des octets émis en éliminant les identificateurs redondants de source et d'objet, qui peuvent être déterminés à partir de l'APDU la plus récemment analysée contenue au sein de la même concaténation d'APDU.

L'adressage d'objet inféré ne doit pas être indiqué dans la première APDU d'une concaténation.

NOTE Toute APDU indiquant le mode d'adressage d'objet inféré dans la première APDU rencontrée dans l'analyse d'ASL est considérée comme étant une APDU mal formée.

Un exemple est inclus dans le Tableau 313.

Tableau 313 – Exemple de cas d'utilisation d'adressage inféré

APDU_1	APDU_2	APDU_3	APDU_4	APDU_5
00 mode d'adressage d'identificateur d'objet	11 mode d'adressage d'identificateur d'objet (indique l'utilisation des OID source et destination issus de l'APDU_1)	11 mode d'adressage d'identificateur d'objet (indique l'utilisation des OID source et destination issus de l'APDU_2, qui utilise les OID source et destination issus de l'APDU_1)	01 mode d'adressage d'identificateur d'objet	11 mode d'adressage d'identificateur d'objet (indique l'utilisation des OID source et destination issus de l'APDU_4)
APDU_1 inclut: - 00 mode d'adressage; - type de service; - identificateur d'objet source de 4 bits; - identificateur d'objet de destination de 4 bits; charge utile spécifique à un service	APDU_2 inclut: - 11 mode d'adressage; - type de service; - charge utile spécifique à un service	APDU_3 inclut: - 11 mode d'adressage; - type de service; - charge utile spécifique à un service	APDU_4 inclut: - 01 mode d'adressage; - type de service; - identificateur d'objet source de 8 bits; - identificateur d'objet de destination de 8 bits; charge utile spécifique à un service	APDU_5 inclut: - 11 mode d'adressage; - type de service; - charge utile spécifique à un service

Le Tableau 314 décrit la construction de l'en-tête d'APDU en mode d'adressage inféré.

Tableau 314 – Construction d'en-tête d'APDU de mode d'adressage inféré

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	Type de primitive de service	1	1	Type de service d'ASL				

12.22.2.5 Codage d'attribut d'objet

12.22.2.5.1 Généralités

Le codage d'attribut d'objet est déterminé par un format d'identificateur d'attribut. Le format peut indiquer:

- Six bits, non indexé: L'attribut s'inscrit dans un entier Unsigned6 et n'est pas indexé.
- Six bits, à un seul indice: L'attribut s'inscrit dans un entier Unsigned6 et exige un seul indice. L'indice de l'attribut est extensible, comme indiqué par le premier bit de l'indice. Si le premier bit de l'indice est 0, l'indice a une taille de 7 bits. Si le premier bit de l'indice est 1, l'indice a une taille de 15 bits.
- Six bits, à deux indices: L'attribut s'inscrit dans un entier Unsigned6 et exige deux indices. Les indices d'attribut sont extensibles individuellement; à savoir, le premier indice peut avoir une taille de 7 bits ou de 15 bits et le second indice peut aussi avoir une taille soit de 7 bits, soit de 15 bits. La taille de l'indice est déterminée par le premier bit de l'indice. Si le premier bit de l'indice est 0, l'indice a une taille de 7 bits. Si le premier bit de l'indice est 1, l'indice a une taille de 15 bits.
- Douze bits, non indexé: L'attribut ne s'inscrit pas dans un entier Unsigned12. L'attribut n'est pas indexé.
- Douze bits, à un seul indice: L'attribut s'inscrit dans un entier Unsigned12 et exige un seul indice. L'indice de l'attribut est extensible, comme indiqué par le premier bit de l'indice. Si le premier bit de l'indice est 0, l'indice a une taille de 7 bits. Si le premier bit de l'indice est 1, l'indice a une taille de 15 bits.
- Douze bits, à deux indices: L'attribut s'inscrit dans un entier Unsigned12 et exige deux indices. Les indices d'attribut sont extensibles individuellement; à savoir, le premier indice peut avoir une taille de 7 bits ou de 15 bits et le second indice peut aussi avoir une taille soit de 7 bits, soit de 15 bits. La taille de l'indice est déterminée par le premier bit de l'indice. Si le premier bit de l'indice est 0, l'indice a une taille de 7 bits. Si le premier bit de l'indice est 1, l'indice a une taille de 15 bits.

NOTE Se référer à 12.23.1.3 pour les définitions de Unsigned6 et de Unsigned12.

12.22.2.5.2 Identificateur d'attribut de 6 bits, non indexé

Le Tableau 315 indique le codage pour un identificateur d'attribut de 6 bits qui n'est pas un attribut indexé ou structuré.

Tableau 315 – Identificateur d'attribut de 6 bits, non indexé

Nombre d'octets	Nombre d'octets							
	7	6	5	4	3	2	1	0
1	Forme courte d'attribut (valeur = 00 binaire)		Identificateur de l'attribut					

12.22.2.5.3 Formes d'identificateur d'attribut de 6 bits, à un seul indice

Le Tableau 316 et le Tableau 317 indiquent le codage pour un identificateur d'attribut de 6 bits qui peut être accessible en utilisant un seul indice.

Tableau 316 – Identificateur d'attribut de 6 bits, à un seul indice, avec un indice de 7 bits

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	Forme courte d'attribut (valeur = 01 binaire)		Identificateur de l'attribut					
1	0	Index						

Tableau 317 – Identificateur d'attribut de 6 bits, à un seul indice, vec un indice de 15 bits

Nombre d'octets	bits							
	7	6	5	4	3	2	1	0
1	Forme courte d'attribut (valeur = 01 binaire)		Identificateur de l'attribut					
2	1	Index (7 bits de poids fort)						
	Index (8 bits de poids faible)							

12.22.2.5.4 Formes d'identificateur d'attribut de 6 bits, à deux indices

Le Tableau 318, le Tableau 319, le Tableau 320 et le Tableau 321 indiquent le codage pour un identificateur d'attribut de 6 bits qui peut être accessible en utilisant deux indices.

Tableau 318 – Identificateur d'attribut de 6 bits, à deux indices, avec deux indices de 7 bits

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	Forme courte d'attribut (valeur = 10 binaire)		Identificateur de l'attribut					
1	0	Index 1						
1	0	Index 2						

Tableau 319 – Identificateur d'attribut de 6 bits, à deux indices, avec deux indices de 15 bits

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	Forme courte d'attribut (valeur = 10 binaire)		Identificateur de l'attribut					
2	1	Index 1 (7 bits de poids fort)						
	Index 1 (8 bits de poids faible)							
2	1	Index 2 (7 bits de poids fort)						
	Index 2 (8 bits de poids faible)							

Tableau 320 – Identificateur d'attribut de 6 bits, à deux indices, avec le premier indice ayant une longueur de 7 bits et le second indice une longueur de 15 bits

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	Forme courte d'attribut (valeur = 10 binaire)		Identificateur de l'attribut					
1	0	Index 1:						
2	1	Index 2 (7 bits de poids fort)						
	Index 2 (8 bits de poids faible)							

Tableau 321 – Identificateur d'attribut de 6 bits, à deux indices, avec le premier indice ayant une longueur de 15 bits et le second indice une longueur de 7 bits

Nombre d'octets	bits							
	7	6	5	4	3	2	1	0
1	Forme courte d'attribut (valeur = 10 binaire)		Identificateur de l'attribut					
2	1	Index 1 (7 bits de poids fort)						
	Index 1 (8 bits de poids faible)							
1	0	Index 2						

12.22.2.5.5 Identificateur d'attribut de 12 bits, non indexé

Le Tableau 322 indique le codage pour un identificateur d'attribut de 12 bits qui n'est pas indexé.

Tableau 322 – Identificateur d'attribut de 12 bits, non indexé

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
2	Forme courte d'attribut (valeur = 11 binaire)		Forme longue d'attribut, forme d'indice= 00 binaire		Identificateur de l'attribut (4 bits de poids fort)			
	Identificateur de l'attribut (8 bits de poids faible)							

12.22.2.5.6 Formes d'identificateur d'attribut de 12 bits, codage à un seul indice

Le Tableau 323 et le Tableau 324 indiquent le codage pour un identificateur d'attribut de 12 bits qui est atteint en utilisant un seul indice.

Tableau 323 – Identificateur d'attribut de 12 bits, à un seul indice, avec un indice de 7 bits

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
2	Forme courte d'attribut (valeur = 11 binaire)		Forme longue d'attribut, forme d'indice= 01 binaire		Identificateur de l'attribut (4 bits de poids fort)			
	Identificateur de l'attribut (8 bits de poids faible)							
1	0	Index						

Tableau 324 – Identificateur d'attribut de 12 bits, à un seul indice, avec un indice de 15 bits

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
2	Forme courte d'attribut (valeur = 11 binaire)		Forme longue d'attribut, forme d'indice= 01 binaire		Identificateur de l'attribut (4 bits de poids fort)			
	Identificateur de l'attribut (8 bits de poids faible)							
2	1	Index 1 (7 bits de poids fort)						
	Index 1 (8 bits de poids faible)							

12.22.2.5.7 Formes d'identificateur d'attribut de 12 bits, codage à deux indices

Le Tableau 325, le Tableau 326, le Tableau 327 et le Tableau 328 indiquent le codage pour un identificateur d'attribut de 12 bits qui est atteint en utilisant deux indices.

Tableau 325 – Identificateur d'attribut de douze bits, à deux indices, avec deux indices de 7 bits

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
2	Forme longue d'attribut (valeur = 11 binaire)		Forme longue d'attribut, forme d'indice = 10 binaire		Identificateur de l'attribut (4 bits de poids fort)			
	Identificateur de l'attribut (8 bits de poids faible)							
1	0	Index 1:						
1	0	Index 2						

Tableau 326 – Identificateur d'attribut de douze bits, à deux indices, avec deux indices de 15 bits

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
2	Forme courte d'attribut (valeur = 11 binaire)		Forme longue d'attribut, forme d'indice= 10 binaire		Identificateur de l'attribut (4 bits de poids fort)			
	Identificateur de l'attribut (8 bits de poids faible)							
2	1	Index 1 (7 bits de poids fort)						
	Index 1 (8 bits de poids faible)							
2	1	Index 2 (7 bits de poids fort)						
	Index 2 (8 bits de poids faible)							

Tableau 327 – Identificateur d'attribut de 12 bits, à deux indices avec le premier indice ayant une longueur de 7 bits et le second indice une longueur de 15 bits

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
2	Forme courte d'attribut (valeur = 11 binaire)		Forme longue d'attribut, forme d'indice= 10 binaire		Identificateur de l'attribut (4 bits de poids fort)			
	Identificateur de l'attribut (8 bits de poids faible)							
1	0	Index 1						
2	1	Index 2 (7 bits de poids fort)						
	Index 2 (8 bits de poids faible)							

Tableau 328 – Identificateur d'attribut de 12 bits, à deux indices avec le premier indice ayant une longueur de 15 bits et le second indice une longueur de 7 bits

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
2	Forme courte d'attribut (valeur = 11 binaire)		Forme longue d'attribut, forme d'indice= 10 binaire		Identificateur de l'attribut (4 bits de poids fort)			
	1	Index 1 (7 bits de poids fort)						
2	Index 2 (8 bits de poids faible)							
	0	Index 2						

12.22.2.5.8 Réservée pour un usage futur

Le Tableau 329 identifie une forme d'identificateur d'attribut qui est réservée pour usage futur.

Tableau 329 – Forme d'identificateur d'attribut de 12 bits, réservée

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	Forme courte d'attribut (valeur = 11 binaire)		Forme longue d'attribut, forme d'indice (réservée): 11 binaire		Réservée pour un usage futur			

12.22.2.6 Lecture

Le Tableau 330 fournit des règles de codage pour la partie spécifique à un service d'une APDU de demande de service de lecture.

L'Application Request ID est un identificateur qui permet au client d'apparier une réponse de service avec la demande originale de service. Une réponse de service doit inclure une copie de l'ID de demande issu de la demande de service correspondante.

Tableau 330 – Règles de codage pour la demande de service de lecture

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	ID de demande							
...	Identificateur de l'attribut (voir les règles de codage pour identificateur d'attribut)							

Le Tableau 331 fournit les règles de codage pour une réponse de service de lecture avec un champ longueur de 7 bits.

Tableau 331 – Règles de codage pour une réponse de service de lecture avec un champ d'une taille de 7 bits

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	ID de demande							
1	Réservé pour usage futur par la présente norme. A des fins de conformité avec cette version de la présente norme, ces bits doivent être définis à 0							Echo de contrôle explicite d'encombrement vers l'avant
1	ServiceFeedbackCode							
1	0	S=Size – incluse conditionnellement seulement lorsque ServiceFeedbackCode indique un succès						
S	Value – conditionnellement présente seulement lorsque ServiceFeedbackCode indique un succès							

NOTE Se référer à 12.23.3 pour les définitions de ServiceFeedbackCode pour les services AL.

Le Tableau 332 fournit les règles de codage pour une réponse de service de lecture avec un champ longueur de 15 bits.

Tableau 332 – Règles de codage pour une réponse de service de lecture avec un champ d'une taille de 15 bits

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	ID de demande							
1	Réservé pour usage futur par la présente norme. A des fins de conformité avec cette version de la présente norme, ces bits doivent être définis à 0							Echo de contrôle explicite d'encombrement vers l'avant
1	ServiceFeedbackCode							
2	1	S[14..8]=Size – 7 bits de poids fort, conditionnellement présente seulement lorsque ServiceFeedbackCode indique un succès						
	S[7..0]= Size – 8 bits de poids faible, conditionnellement présente seulement lorsque ServiceFeedbackCode indique un succès							
S	Value – conditionnellement présente seulement lorsque ServiceFeedbackCode indique succès							

12.22.2.7 Ecriture

Le Tableau 333 et le Tableau 334 fournissent les règles de codage pour une demande de service d'écriture.

L'Application Request ID est un identificateur qui permet au client d'apparier une réponse de service avec la demande originale de service. Une réponse de service doit inclure une copie de l'ID de demande issu de la demande de service correspondante.

Tableau 333 – Règles de codage pour une demande de service d'écriture avec un champ d'une taille de 7 bits

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	ID de demande							
...	Identificateur de l'attribut (voir les règles de codage d'attribut)							
...	0	S=Size						
S	Valeur							

Tableau 334 – Règles de codage pour une demande de service d'écriture avec un champ d'une taille de 15 bits

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	ID de demande							
...	Identificateur de l'attribut (voir les règles de codage d'attribut)							
2	1	S[14..8]==Size (7 bits de poids fort)						
	S[7..0]=Size (8 bits de poids faible)							
S	Valeur							

Le Tableau 335 fournit les règles de codage pour une réponse de service d'écriture.

Tableau 335 – Règles de codage pour la réponse de service d'écriture

Nombre d'octets	bits							
	7	6	5	4	3	2	1	0
1	ID de demande							
1	Réservé pour usage futur par la présente norme. A des fins de conformité avec cette version de la présente norme, ces bits doivent être définis à 0							Echo de contrôle explicite d'encombrement vers l'avant
1	ServiceFeedbackCode							

12.22.2.8 Execute

Le Tableau 336 et le Tableau 337 fournissent les règles de codage pour une demande de service Execute.

L'Application Request ID est un identificateur qui permet au client d'apparier une réponse de service avec la demande originale de service. Une réponse de service doit inclure une copie de l'ID de demande issu de la demande de service correspondante.

Tableau 336 – Règles de codage pour une demande de service Execute avec un champ d'une taille de 7 bits

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	Identificateur de demande							
1	Identificateur de méthode							
1	0	S=Size en octets des paramètres de demande						
S	Paramètres de demande							

Tableau 337 – Règles de codage pour une demande de service Execute avec un champ d'une taille de 15 bits

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	Identificateur de demande							
1	Identificateur de méthode							
2	1	S[14..8]=Size en octets des paramètres de réponse (7 bits de poids fort)						
	S[7..0]=Size (8 bits de poids faible)							
S	Paramètres de réponse							

Le Tableau 338 et le Tableau 339 fournissent les règles de codage pour une réponse de service Execute.

Tableau 338 – Règles de codage pour une réponse de service Execute avec un champ de taille de 7 bits

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	Identificateur de demande							
1	Réservé pour usage futur par la présente norme. A des fins de conformité avec cette version de la présente norme, ces bits doivent être définis à 0							Forward explicit congestion control echo (écho de contrôle explicite d'encombrement vers l'avant)
1	ServiceFeedbackCode							
1	0	S=Taille en octets des paramètres de réponse						
S	Paramètres de réponse							

Tableau 339 – Règles de codage pour une réponse de service Execute avec un champ de taille de 15 bits

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	Identificateur de demande							
2	Réservé pour usage futur par la présente norme. A des fins de conformité avec cette version de la présente norme, ces bits doivent être définis à 0							Forward explicit congestion control echo (écho de contrôle explicite d'encombrement vers l'avant)
1	ServiceFeedbackCode							
2	1	S[14..8]=Taille en octets des paramètres de réponse (7 bits de poids fort)						
	S[7..0]=Taille (8 bits de poids faible)							
S	Paramètres de réponse							

12.22.2.9 Tunnel

Le Tableau 340 et le Tableau 341 fournissent les règles de codage pour une demande de service Tunnel.

Tableau 340 – Règles de codage pour une demande de service Tunnel avec un champ de taille de 7 bits

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	0	S=Taille 7 bits						
S	Payload							

Tableau 341 – Règles de codage pour une demande de service Tunnel avec un champ de taille de 15 bits

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
2	1	S[14..8]=Taille en octets des paramètres de réponse (7 bits de poids fort)						
	S[7..0]=Taille (8 bits de poids faible)							
S	Payload							

Le Tableau 342 et le Tableau 343 fournissent les règles de codage pour une réponse de service Tunnel.

Tableau 342 – Règles de codage pour une réponse de service Tunnel avec un champ de taille de 7 bits

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	Réservé pour usage futur par la présente norme. A des fins de conformité avec cette version de la présente norme, ces bits doivent être définis à 0							Forward explicit congestion control echo (écho de contrôle explicite d'encombrement vers l'avant)
1	0	S=Taille						
S	Payload							

Tableau 343 – Règles de codage pour une réponse de service Tunnel avec un champ de taille de 15 bits

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	Réservé pour usage futur par la présente norme. A des fins de conformité avec cette version de la présente norme, ces bits doivent être définis à 0							Forward explicit congestion control echo (écho de contrôle explicite d'encombrement vers l'avant)
2	1	S[14..8]=Taille en octets des paramètres de réponse (7 bits de poids fort)						
	S[7..0]=Taille (8 bits de poids faible)							
S	Payload							

12.22.2.10 AlertReport

Le Tableau 344 et le Tableau 345 fournissent les règles de codage pour une demande de service AlertReport.

Tableau 344 – Règles de codage pour le service AlertReport avec un champ de taille de 7 bits de données associées

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	Alert report ID (ID de rapports d'alertes)							
2	Identificateur de processus d'application d'objet de détection (port T)							
2	Identificateur d'objet de détection							
6	TAINetworkTime							
1	Classe	Direction	Catégorie		Priorité d'alerte			
1	Type							
1	0	S=Taille des données associées						
S	Données associées							

Tableau 345 – Règles de codage pour le service AlertReport avec un champ de taille de 15 bits de données associées

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	Alert report ID (ID de rapports d'alertes)							
2	Identificateur de processus d'application d'objet de détection (port T)							
2	Identificateur d'objet de détection							
6	TAINetworkTime							
1	Classe	Direction	Catégorie		Priorité d'alerte			
1	Type							
2	1	S[14..8]=Taille en octets des paramètres de réponse (7 bits de poids fort)						
	S[7..0]=Taille (8 bits de poids faible)							
S	Données associées							

12.22.2.11 AlertAcknowledge

Le Tableau 346 fournit les règles de codage d'une demande de service AlertAcknowledge.

Tableau 346 – Règles de codage pour le service AlertAcknowledge

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	Alert report ID (ID de rapports d'alertes)							

12.22.2.12 Publish

Le Tableau 347 fournit les règles de codage pour un service Publish natif.

Lorsqu'il est utilisé conjointement avec un objet concentrateur, le champ Data dans la charge utile comprend les données entières communiquées, qui sont une séquence configurée des variables de contrôle de processus. Les variables de contrôle de processus incluent à la fois les informations de statut et les valeurs de processus. La structure des données est indiquée par la version du contenu d'édition. Le numéro de séquence se situe dans le domaine d'application d'un objet concentrateur particulier.

Tableau 347 – Règles de codage pour le service Publish pour une séquence native de valeurs

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
1	Publishing content version							
1	Numéro de séquence de fraîcheur							
S	Data							

Le Tableau 348 fournit les règles de codage pour un service Publish utilisé pour acheminer soit une chaîne d'octets codés en interne, soit des données non natives. L'utilisation de ce service pour des données non natives permet la prise en charge de la tunnellisation.

Tableau 348 – Règles de codage pour le service Publish – non natif (prise en charge de tunnel)

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
S	Data							

Les règles de codage pour les données non interprétées de différentes tailles, données au Tableau 351, s'appliquent à un healthReport publié (voir 12.23.1.6).

12.22.2.13 Concaténation

Le Tableau 349 fournit les règles de codage pour construire une seule TSDU qui contient plusieurs APDU logiques.

Tableau 349 – Règles de codage pour le service Concatenate

Nombre d'octets	Bits							
	7	6	5	4	3	2	1	0
S	SEQUENCE OF APDUs							

12.22.3 Codage des données d'application

12.22.3.1 Généralités

Le codage des éléments de données d'application uniques est toujours primitif. Dans les tableaux de 12.22.3, l'octet 1 représente l'octet de poids fort, le bit 7 représente le bit de poids fort au sein d'un octet, et le bit 0 représente le bit de poids faible au sein d'un octet.

La sémantique des données d'utilisateur est connue par:

- accord préalable (contenu de la charge utile de tunnel, par exemple);
- position dans l'APDU avec une taille de champs fixe pour le contenu; ou
- champs existants dans l'APDU.

Dans ces situations, aucune information de décodage complémentaire n'est ajoutée à l'APDU.

Les règles de codage pour les données d'application sont fournies dans le Tableau 350 et dans le Tableau 351. Si la taille est fixe, comme dans le cas du type de données OctetStringN pour une valeur fixe donnée de N, alors les informations de taille sont implicites dans la déclaration, elles ne sont pas acheminées explicitement dans l'APDU, comme indiqué dans le Tableau 350.

Tableau 350 – Règle de codage générale pour les données d'application de taille invariable

Data (taille fixe)

En contraste, si la taille peut changer, comme dans le cas du type de données (et pas OctetStringN pour n'importe quel N), la taille du champ réel est explicitement couverte dans l'APDU. Cela est souvent fait en préfixant les données avec la taille, comme illustré au Tableau 351. Dans d'autres cas, le champ size est trouvé directement dans ou est calculable depuis un champ trié précédemment dans l'APDU.

Tableau 351 – Règles de codage pour les données d'application de taille variable de 0..255 octets

Unsigned8 N, taille des données (en octets)	Data (taille N octets)

Les paragraphes 12.22.3.2 à 12.22.3.8 définissent le codage des données pour les types de données normalisés.

12.22.3.2 Valeurs booléennes

NOTE Le nom de ce type a été attribué en l'honneur du logicien George Boole.

12.22.3.2.1 Codage de valeurs booléennes

Les booléens sont codés sous la forme de valeurs égales à zéro/différentes de zéro dans un champ 1 bit pour les structures de données condensées ou 8 bits pour les structures de données relativement non condensées.

12.22.3.2.2 Boolean8

Le codage Boolean8, utilisé dans des structures de données relativement non condensées, est:

- Data type: Boolean
- Taille: 1 octet

Une valeur égale à zéro de la représentation Unsigned8 sous-jacente code la valeur false; une valeur non égale à zéro code la valeur true.

12.22.3.2.3 Boolean1

Le codage Boolean1, utilisé dans des structures de données condensées, est:

- Data type: Boolean
- Taille: 1 bit

Une valeur égale à zéro de la représentation Unsigned1 sous-jacente code la valeur false; une valeur non égale à zéro égale à un (1) code la valeur true.

12.22.3.3 Valeurs entières

12.22.3.3.1 Codage de valeurs entières signées

12.22.3.3.1.1 Généralités

Les entiers signés sont codés sous forme de nombre de complément à 2. Dans la représentation de complément à 2, les nombres négatifs sont représentés par le complément à 2 de la valeur absolue. Dans ce système, zéro a une seule représentation.

Dans la représentation de complément à 2, les nombres positifs sont représentés comme un binaire simple, et les nombres en complément à 2 négatifs sont représentés comme le nombre binaire qui, lorsqu'il est ajouté à un nombre positif de même grandeur, est égal à zéro.

Le bit de poids fort (à savoir, le bit 7 pour une valeur Integer8, le bit 15 pour un Integer16) indique le signe du nombre entier, et s'appelle donc le bit de signe. Si le bit de signe est zéro, le nombre représenté est supérieur ou égal à zéro (à savoir zéro ou un nombre positif). Si le bit de signe est un, le nombre représenté est inférieur à zéro (à savoir un nombre négatif).

NOTE Pour calculer le complément à 2 d'un nombre entier, on inverse l'équivalent binaire du nombre en changeant tous les uns en zéros et tous les zéros en uns (également appelé complément à 1) et ensuite on ajoute un.

Exemple: Formation du complément à 2 de la valeur 17.

0x 0001 000 1 (17 en binaire)

Pour former le complément à 2:

En premier lieu: NOT (0x 0001 000 1) = 0x 1110 111 0, où l'opération NOT provoque l'inversion des bits.

Ensuite, on ajoute 1: (0x 1110 111 0) + (0x 0000 0001) = 0x 1110 1111 (complément à deux = -17).

12.22.3.3.1.2 Integer8

Le codage d'un Integer8 est:

- Data type: Integer8
- Plage: $-2^7 \leq k \leq 2^7 -1$ (c'est-à-dire, $-128 \leq k \leq 127$)
- Taille: 1 octet

12.22.3.3.1.3 Integer16

Le codage d'un Integer16 est:

- Data type: Integer16
- Plage: $-2^{15} \leq k \leq 2^{15} -1$ (c'est-à-dire, $-32\,768 \leq k \leq 32\,767$)
- Taille: 2 octets

12.22.3.3.1.4 Integer32

Le codage d'un Integer32 est:

- Data type: Integer32
- Plage: $-2^{31} \leq k \leq 2^{31} -1$ (c'est-à-dire, $-2\,147\,483\,648 < k < 2\,147\,483\,647$)
- Taille: 4 octets

12.22.3.3.1.5 IntegerN

Le codage IntegerN, utilisé dans des structures de données condensées, est:

- Data type: IntegerN
- Plage: $-2^{-(N-1)} \leq k \leq 2^{(N-1)}-1$
- Taille: N bits

12.22.3.3.2 Codage de valeurs entières non signées

12.22.3.3.2.1 Unsigned8

Le codage d'un Unsigned8 est:

- Data type: Unsigned8
- Plage: $0 \leq k \leq 2^8 -1$ (c'est-à-dire, $0 \leq k \leq 255$)
- Taille: 1 octet

Le Tableau 352 fournit les règles de codage pour les données Unsigned8.

Tableau 352 – Règles de codage pour Unsigned8

Octet	Bits							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

12.22.3.3.2.2 Unsigned16

Le codage d'un Unsigned16 est:

- Data type: Unsigned16
- Plage: $0 \leq k \leq 2^{16}$ -1 (c'est-à-dire, $0 \leq k \leq 65\,535$)
- Taille: 2 octets

Le Tableau 353 fournit les règles de codage pour les données Unsigned16.

Tableau 353 – Règles de codage pour Unsigned16

Octet	Bits							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8
2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

12.22.3.3.2.3 Unsigned32

Le codage d'un Unsigned32 est:

- Data type: Unsigned32
- Plage: $0 \leq k \leq 2^{32}$ -1 (c'est-à-dire, $0 \leq k \leq 4\,294\,967\,295$)
- Taille: 4 octets

Le Tableau 354 fournit les règles de codage pour les données Unsigned32.

Tableau 354 – Règles de codage pour Unsigned32

Octet	Bits							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8
4	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

12.22.3.3.2.4 Unsigned64

Le codage d'un Unsigned64 est:

- Data type: Unsigned64
- Taille: 8 octets
- Plage: $0 \leq k \leq 2^{64}$ -1 (c'est-à-dire, $0 \leq k \leq 18\,446\,744\,073\,709\,551\,615$)

Le Tableau 355 fournit les règles de codage pour les données Unsigned64.

Tableau 355 – Règles de codage pour Unsigned64

Octet	Bits							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	2^{63}	2^{62}	2^{61}	2^{60}	2^{59}	2^{58}	2^{57}	2^{56}
2	2^{55}	2^{54}	2^{53}	2^{52}	2^{51}	2^{50}	2^{49}	2^{48}
3	2^{47}	2^{46}	2^{45}	2^{44}	2^{43}	2^{42}	2^{41}	2^{40}
4	2^{39}	2^{38}	2^{37}	2^{36}	2^{35}	2^{34}	2^{33}	2^{32}
5	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
6	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
8	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

12.22.3.3.2.5 Unsigned128

Le codage d'un Unsigned128 est:

- Data type: Unsigned128
- Taille: 16 octets
- Plage: $0 \leq k \leq 2^{128}-1$ (c'est-à-dire, $0 \leq k \leq$ 340 282 366 920 938 463 463 374 607 431 768 211 455)

Le Tableau 356 fournit les règles de codage pour les données Unsigned128.

Tableau 356 – Règles de codage pour Unsigned128

Octet	Bits							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	2^{127}	2^{126}	2^{125}	2^{124}	2^{123}	2^{122}	2^{121}	2^{120}
2	2^{119}	2^{118}	2^{117}	2^{116}	2^{115}	2^{114}	2^{113}	2^{112}
3	2^{111}	2^{110}	2^{109}	2^{108}	2^{107}	2^{106}	2^{105}	2^{104}
4	2^{103}	2^{102}	2^{101}	2^{100}	2^{99}	2^{98}	2^{97}	2^{96}
5	2^{95}	2^{94}	2^{93}	2^{92}	2^{91}	2^{90}	2^{89}	2^{88}
6	2^{87}	2^{86}	2^{85}	2^{84}	2^{83}	2^{82}	2^{81}	2^{80}
7	279	2^{78}	2^{77}	2^{76}	2^{75}	2^{74}	2^{73}	2^{72}
8	2^{71}	2^{70}	2^{69}	2^{68}	2^{67}	2^{66}	2^{65}	2^{64}
9	2^{63}	2^{62}	2^{61}	2^{60}	2^{59}	2^{58}	2^{57}	2^{56}
10	2^{55}	2^{54}	2^{53}	2^{52}	2^{51}	2^{50}	2^{49}	2^{48}
11	2^{47}	2^{46}	2^{45}	2^{44}	2^{43}	2^{42}	2^{41}	2^{40}
12	2^{39}	2^{38}	2^{37}	2^{36}	2^{35}	2^{34}	2^{33}	2^{32}
13	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
14	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
15	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
16	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

12.22.3.3.2.6 UnsignedN

Le codage UnsignedN, utilisé dans des structures de données condensées, est:

- Data type: UnsignedN
- Plage: $0 \leq k \leq 2^{N)}$-1
- Taille: N bits

12.22.3.4 Valeurs de virgule flottante

12.22.3.4.1 Codage de valeurs en virgule flottante

La présente norme utilise le codage défini par l'ISO/IEC/IEEE 60559 (basé sur l'IEEE 754) pour les valeurs à virgule flottante normalisées et les NaN. Chaque valeur est représentée par trois champs contigus:

- S, le signe de la valeur en virgule flottante, où 0 et 1 représentent respectivement le positif et le négatif, acheminé dans un champ de 1-bit;
- E, l'exposant de la valeur, en base 2, biaisé de B, acheminé dans un champ de N_E = environ 1/4 du nombre total de bits de la représentation de la valeur à virgule flottante, où la valeur de B est $2^{(N_E-1)}$-1;
- F, la partie fractionnaire de la mantisse de la valeur, également en base 2, acheminée dans les N_F bits restants de la représentation de la valeur.

Quand E n'est pas composé que de bits égaux à zéro ou que de bits égaux à un, la valeur numérique résultante est $(-1)^S \times 2^{(E-B)} \times (1,F)$. Quand E et F sont tous deux égaux à zéro bits, la valeur représentée est un zéro signé.

Quand E ne comporte que des bits un et F que des bits zéro, la valeur représentée est un infini signé. Voir l'ISO/IEC/IEEE 60559 pour plus d'informations sur la représentation, la plage et la précision des nombres réels, y compris le codage du zéro signé, de l'infini signé (débordement), des nombres dénormalisés (soupassement) et des NaN.

12.22.3.4.2 Single-precision float (virgule flottante en simple précision)

Les valeurs en virgule flottante en simple précision sont représentées de façon contiguë comme indiqué dans le Tableau 357, où N_E = 8, B = 127 et N_F = 23. Cela permet à une valeur en virgule flottante en simple précision d'être calculée par l'équation suivante, qui s'applique lorsque E n'est pas des zéros partout ou des uns partout:

$$(-1)^S \times 2^{(E - 127)} \times (1,F)$$

Tableau 357 – Règles de codage pour single-precision float

Octet	Bits							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	Signe (S)	Exposant (E)						
1	+/-	2^7	2^6	2^5	2^4	2^3	2^2	2^1
	(E)	Fraction (F)						
2	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
3	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}
4	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}	2^{-21}	2^{-22}	2^{-23}

12.22.3.5 Double-precision float (virgule flottante en double précision)

Les valeurs en virgule flottante en double précision sont représentées de façon contiguë comme indiqué dans le Tableau 358, où N_E = 11, B = 1 023 et N_F = 52. Cela permet à une valeur en virgule flottante en double précision d'être calculée par l'équation suivante, qui s'applique lorsque E n'est pas des zéros partout ou des uns partout:

$(-1)^S \times 2^{(E - 1\,023)} \times (1,F)$

Tableau 358 – Règles de codage pour double-precision float

Octet	Bits							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	Signe (S)	Exposant (E)						
	+/-	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4
2	Exposant (E) (suite)				Fraction (F)			
	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
3	Fraction (F) (suite)							
	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}
4	2^{-13}	2^{-14}	2^{-15}	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}
5	2^{-21}	2^{-22}	2^{-23}	2^{-24}	2^{-25}	2^{-26}	2^{-27}	2^{-28}
6	2^{-29}	2^{-30}	2^{-31}	2^{-32}	2^{-33}	2^{-34}	2^{-35}	2^{-36}
7	2^{-37}	2^{-38}	2^{-39}	2^{-40}	2^{-41}	2^{-42}	2^{-43}	2^{-44}
8	2^{-45}	2^{-46}	2^{-47}	2^{-48}	2^{-49}	2^{-50}	2^{-51}	2^{-52}

12.22.3.6 VisibleString

Le codage d'une chaîne visible est:

- Type: VisibleString
- Plage: Voir ISO/IEC 646 et ISO/IEC 2375: Définition de numéro d'enregistrement 2 + ESPACE
- Codage: Codage: Voir l'ISO 646

NOTE Voir l'ISO/IEC 2375 pour plus de détails.

Le Tableau 359 fournit les règles de codage pour les données VisibleString. Si la taille de la chaîne n'est pas déterminable à partir d'autres facteurs, la taille en octets est codée dans un octet qui précède immédiatement la chaîne OctetString, comme spécifié en Tableau 351.

Tableau 359 – Règles de codage pour VisibleString

Octet	Bits							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	Premier caractère de la chaîne							
2	Deuxième caractère de la chaîne							
…	….							
N	Dernier caractère de la chaîne							

12.22.3.7 OctetString

Le codage d'une chaîne octet est:

- Type: OctetString
- Codage: Binaire

Le Tableau 360 fournit les règles de codage pour les données OctetString. Si la taille de la chaîne n'est pas déterminable à partir d'autres facteurs, la taille en octets est codée dans un octet qui précède immédiatement la chaîne OctetString, comme spécifié en Tableau 351.

Tableau 360 – Règles de codage pour OctetString

Octet	Bits							
	7	6	5	4	3	2	1	0
1	Premier octet de la chaîne							
…	…							
N	Dernier octet de la chaîne							

12.22.3.8 BitString

Le codage d'une chaîne de bits qui ne fait pas partie d'une structure condensée supérieure est:

- Type: BitString
- Codage: Binaire
- Taille: Seuls les multiples de 8 bits (à savoir: les multiples d'octets) sont pris en charge pour BitString sans faire partie d'une structure condensée supérieure

Le Tableau 361 fournit les règles de codage générales pour les données BitString. Si la taille de la chaîne n'est pas déterminable à partir d'autres facteurs, la taille en octets est codée dans un octet qui précède immédiatement la chaîne BitString, comme spécifié au Tableau 351.

Tableau 361 – Règles de codage pour BitString

Octet	Bits							
	7	6	5	4	3	2	1	0
1	(8 x *N*-1)ème	(8 x *N*-2)ème	(8 x *N*-3)ème	(8 x *N*-4)ème	(8 x *N*-5)ème	(8 x *N*-6)ème	(8 x *N*-7)ème	(8 x *N*-8)ème
	(position du bit dans la chaîne)							
2	(8 x *N*-9)ème	(8 x *N*-10)ème	(8 x *N*-11)ème	(8 x *N*-12)ème	(8 x *N*-13)ème	(8 x *N*-14)ème	(8 x *N*-15)ème	(8 x *N*-16)ème
…								
N	etc.							

12.22.3.9 Clé symétrique (SymmetricKey)

Une clé symétrique est opaque. Dans cette édition de la présente norme, il s'agit d'une clé de 128 bits. En tant que telle, elle est mise en correspondance, sans interprétation, avec un Octet16 qui mesure seize octets.

12.22.4 Types de données relatives au temps

12.22.4.1 Généralités

Le temps est continu et potentiellement représenté avec une précision presque infinie dans une plage presque infinie. Ainsi, toute représentation raisonnable du temps a une résolution finie spécifique (par exemple 1 h, 1 s, 1 ns, 10^{-20} s, etc.) et une plage spécifique, telle que [0..1 j) ou (0..10 000 ans], et toute valeur temporelle doit être représentée par le modulo de ces valeurs.

Dans la présente norme, TAINetworkTime est représenté avec une résolution de 2^{-16} s et une plage de [0..2^{32}) s, modulo 2^{32} s. TAITimeRounded a la même plage mais est arrondi à la seconde la plus proche et a une résolution de 2^{0} s (c'est-à-dire 1 s).

TAITimeDifference est conçu pour représenter la différence entre deux valeurs différentes de TAINetworkTime. Cette différence est également représentée modulo 2^{32} s, de sorte que des valeurs numériques très importantes représentent des différences négatives. L'identification de la partie de la plage de 2^{32} s d'une valeur TAITimeDifference qui est interprétée comme différence positive et de la partie qui est interprétée comme différence négative est déterminée par l'utilisation qui est faite de cette différence.

EXEMPLE Lors de la différenciation de deux valeurs TAINetworkTime au cours du traitement d'un nonce TPDU, la logique indiquée adresse spécifiquement les différences comprises dans une petite plage signée et catégorise toutes les autres différences comme étant "hors plage" sans essayer de les assigner au passé ou au futur relativement éloignés du temps TAI référencé.

12.22.4.2 TAINetworkTime

TAINetworkTime représente le temps du réseau en temps TAI sous la forme d'une valeur binaire à point fixe de six octets avec une résolution of 2^{-16} s modulo 2^{32} s. Ainsi les quatre octets supérieurs représentent le temps TAI actuel en unités de 1 s tandis que les deux octets inférieurs représentent le temps TAI fractionnel en unités de 2^{-16} s.

- Data type: TAINetworkTime

NOTE 1 Cette représentation s'applique aussi à TAITimeDifference qui est une différence modulo.

- Plage valide exprimée en tant que valeur binaire à point fixe non signée
  - dont le composant intégral a la plage 0..2^{32}-1 s (modulo 2^{32} s);
  - et dont le composant fractionnel a une résolution de 2^{-16} s.

NOTE 2 Etant donné que toutes les valeurs possibles se produisent de façon répétée (cycliquement) dans une représentation modulo telle que TAINetworkTime, il n'est pas possible de coder des valeurs à signification spéciale dans cette plage comme cela peut être fait avec les extrémités d'une plage linéaire.

Tableau 362 montre la représentation de TAINetworkTime et de TAITimeDifference lors de l'interprétation comme différence modulo.

Tableau 362 – Règles de codage pour TAINetworkTime et TAITimeDifference lors de l'interprétation comme différence modulo

Octet	Bits								Interprétation
	7	6	5	4	3	2	1	0	
1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Partie intégrale du temps TAI avec une granularité de 1 s
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
5	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	Partie fractionnelle du temps TAI avec une granularité de 2^{-16} s
6	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}	2^{-16}	

12.22.4.3 TAITimeDifference

Le codage pour TAITimeDifference est identique à celui de TAINetworkTime. Cependant, puisqu'il s'agit d'une valeur modulo, elle a des interprétations potentielles en tant que valeurs signées. Ces interprétations sont:

- Data type: TAITimeDifference
- Plage valide exprimée en tant que valeur binaire à point fixe non signée à 2 compléments
  - dont le composant intégral a la plage -2^{32}..2^{32}-1 s;

– dont le composant fractionnel a une résolution de 2^{-16} s; et

– qui est considérée comme "s'enroulant" des valeurs positives aux valeurs négatives à une certaine valeur Unsigned32 pour le composant intégral qui dépend du scénario d'utilisation spécifique.

12.22.4.4 TAITimeRounded

TAITimeRounded représente le temps TAI en secondes intégrales modulo la période de la représentation, arrondi à la seconde la plus proche. Son codage est:

- Data type: TAITimeRounded
- Plage valide: 0..2^{32}-1 s (modulo 2^{32} s)

Le Tableau 363 montre la représentation de TAITimeRounded.

Tableau 363 – Règles de codage pour TAITimeRounded

Octet	Bits								
	7	6	5	4	3	2	1	0	Interprétation
1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Temps TAI avec une granularité de 1 s
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
4	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	

12.22.4.5 Structures de données normalisées

Les structures de données normalisées sont codées en concaténant les valeurs codées pour les éléments de la structure dans l'ordre allant de l'élément de plus petit numéro à l'élément de plus grand numéro, en commençant à l'octet 1 du résultat codé.

12.22.4.6 Null

Le type de données null a une taille de zéro (0) octet. La valeur null est souvent utilisée pour la cohérence sémantique, où elle représente le potentiel pour un contenu lorsqu'aucun contenu n'a été identifié.

12.22.4.7 Packed

Le type de données condensé indique qu'un ou plusieurs éléments des types de données normalisés ont été concaténés ensemble sans trous pour maintenir l'alignement des octets. De plus, les éléments condensés de type BitString et BooleanArray peuvent ne pas occuper un nombre entier d'octets. La structure et la composition des données packed est implicitement connue par les correspondants.

NOTE Les matrices BooleanArray sont généralement représentées sous la forme de chaînes BitString condensées.

12.22.4.8 Données structurées

12.22.4.8.1 SEQUENCE

SEQUENCE est utilisé pour indiquer des données structurées de(s) même(s) ou différent(s) type(s) de données normalisé(s). Cela s'apparente à une construction enregistrement.

La présente norme ne prend pas en charge les séquences qui contiennent les membres facultativement présents. Si un tel besoin est identifié, une séquence (structure) séparée doit être définie pour chaque séquence de membres exigée de cette nature. Les correspondants

doivent avoir une connaissance préalable de la structure de la séquence; donc, aucun mécanisme n'est fourni pour acheminer explicitement sa structure.

12.22.4.8.2 SEQUENCE OF

Pour des données, SEQUENCE OF sert à indiquer une construction ARRAY (matrice). Le contenu de la matrice peut être acheminé dans son intégralité, ou un élément individuel spécifié d'une matrice peut être acheminé.

Pour l'acheminement d'un élément individuel, le type de données de l'élément est implicitement connu par les correspondants. Sachant que certains types de données ont une taille variable, la taille de l'élément est acheminée avec les données de l'élément.

Lorsque des matrices sont acheminées dans leur intégralité, elles sont codées dans l'ordre rangée d'abord. La taille de la matrice en octets doit également être incluse. Le type de données des éléments est également connu implicitement par les points d'extrémité correspondants, et n'est pas explicitement spécifié dans l'APDU. La(les) dimension(s) de la matrice est(sont) également implicitement connue(s) par les points d'extrémité correspondants, et n'est(ne sont) donc pas explicitement incluse(s) dans l'APDU.

NOTE Selon la notation matricielle normalisée, les rangées sont identifiées par le premier indice d'une matrice bidimensionnelle et les colonnes par le deuxième indice. Par exemple, pour le langage de programmation "C", une matrice bidimensionnelle composée de deux lignes et de trois colonnes, ce qui donnerait visuellement

```
1  2  3
4  5  6 '
```

peuvent être définies par

```
int A[2][3] = { {1, 2, 3}, {4, 5, 6} }
```

Le codage de la présente norme transporterait les éléments de cette matrice dans l'ordre ci-dessous: 1, 2, 3, 4, 5, 6.

12.22.4.8.3 CHOICE

CHOICE représente une sélection choisie dans une énumération prédéfinie de possibilités acceptables. Le contenu des données varie en fonction du choix sélectionné.

12.22.4.8.4 OPTIONAL

OPTIONAL indique que le composant désigné peut ne pas être inclus dans la structure contenante.

12.22.4.8.5 IMPLICIT

IMPLICIT indique que les aspects de codage qui identifient le type, la taille, la sélection et la présence ou l'absence en tant qu'élément optionnel doivent être supprimés lorsque ces informations sont déterminables autrement à partir du contexte, par exemple à partir d'autres éléments de la structure de données.

NOTE Lorsqu'une déclaration de type de données se termine avec un nombre entier explicite indiquant la taille du type atomique (par ex: Unsigned12) ou le nombre d'éléments d'un type de matrice (par ex: OctetString2), ce nombre entier est implicite dans la déclaration et n'est pas transporté dans la PDU en tant que taille explicitement acheminée dans l'élément lui-même. Ainsi, un OctetStringN ne contient pas de champ spécifiant N, tandis qu'un OctetString contient un tel champ (car la taille n'est pas implicite dans la déclaration).

12.23 Syntaxe

12.23.1 Unité de données de protocole d'application

12.23.1.1 Début du module conteneur

NOTE 1 La racine d'identificateur d'objet pour les définitions suivantes a été changée en une racine IEC pour prendre en charge la correction et l'évolution de la structure des TSDU par rapport à celle de structure ISA 100.11a TSDU d'origine.

NOTE 2 La déclaration d'extensibilité ASN.1 "..." est utilisée dans chaque production pouvant être étendue dans les éditions futures de la présente norme, ou dans des modes spécifiques à l'industrie ou au fournisseur dans le cadre de cette édition.

```
IEC62734 (1 0 62734) edition (1) TSDU (1) ) DEFINITIONS
      IMPLICIT TAGS
      EXPORTS IEC62734_TSDU;
      ::= BEGIN
```

NOTE 3 Pour cette édition de l'IEC 62734, la structure au niveau des bits de l'IEC62734_TSDU est identique à celle de l'ISA 100_TSDU dans l'ISA 100.11a:2011, 11.23; un des préfixes désigne donc une structure avec une représentation concrète identique et une sémantique associée similaire. La déclaration de préfixe équivalente pour l'ISA 100.11a était

```
--  ( ISA ( ) ISA100.11a:2011 (71) ) DEFINITIONS
--      IMPLICIT TAGS
--      EXPORTS ISA100_TSDU;
--      ::= BEGIN
```

12.23.1.2 Définitions de haut niveau

```
IEC62734_TSDU:: = IMPLICIT CHOICE (
      individualAPDU                ASLIndividualAPDU,
      concatenatedAPDU              ASLConcatenatedAPDU
)
ASLIndividualAPDU::= IMPLICIT CHOICE(
   confirmedRequestAPDU             ASLConfirmedServiceRequest,
   confirmedResponseAPDU            ASLConfirmedServicerResponse,
   unconfirmedRequestAPDU           ASLUnconfirmedServiceRequest,
   publicationAPDU                  ASLPublicationRequest
)
ASLConcatenatedAPDU::= IMPLICIT SEQUENCE (
   IMPLICIT CHOICE (
      -- implicit based on the content of the APDU header, which is common across the choices
      confirmedRequest              ASLConfirmedServiceRequest,
      confirmedResponse             ASLConfirmedServiceResponse,
      unconfirmedRequest            ASLUnconfirmedServiceRequest
   )
)
```

NOTE La concaténation fonctionne, car la taille de chaque APDU apériodique est soit déterminée par des informations explicites, soit est implicite par la définition de primitives de service.

12.23.1.3 Substitutions communes

```
Float32::= REAL (WITH COMPONENTS(, base(2)) SIZE 32)          -- single-precision binary float
Float64::= REAL (WITH COMPONENTS(, base(2)) SIZE 64)          -- double-precision binary float

Integer8::= INTEGER (-128..127)                               -- 8-bit integer
Integer16::= INTEGER(-32 768..32 767)                         -- 16-bit integer
Integer32::= INTEGER(-4 294 967 296..4 294 967 295)           -- 32-bit integer

Unsigned8::= INTEGER(0..255)                                  -- 8-bit unsigned
Unsigned16::= INTEGER(0..65 535)                              -- 16-bit unsigned
Unsigned32::= INTEGER(0..4 294 967 295)                 -- 32-bit unsigned
Unsigned64::= INTEGER(0..18 446 744 073 709 551 615)-- 64-bit unsigned
Unsigned128::= INTEGER(0..340 282 366 920 938 463 374 607 431 768 212 455) -- 128-bit unsigned

Octet1::= Unsigned8
DL16Address::= Unsigned16
EUI64Address::= Unsigned64
IPv6Address::= Unsigned128
SymmetricKey::= PACKED ARRAY [128] OF BIT   -- opaque, uninterpretable bit string
```

```
TAINetworkTime::= SEQUENCE (          -- referenced to TAI start time instant
   Seconds                            Unsigned32,
   FractionalSecond                   Unsigned16
)

TAITimeRounded::= SEQUENCE (          -- referenced to TAI start time instant
   Seconds                            Unsigned32,
)

TAITimeDifference::= SEQUENCE (       -- not referenced to TAI start time instant
   Seconds                            Unsigned32,
   FractionalSecond                   Unsigned16
)                                     -- See NOTE 1
```

NOTE 1 Comme la représentation du temps TAI dans la présente norme est modulo 2^{32} s, une valeur de type TAITimeDifference peut être interprétée comme une différence positive ou négative, avec une différente entre les deux de 2^{32} s. Des utilisations différentes de ce type imposent des limites locales différences dans la plage attendue de différence numérique, qui à leur tour déterminent la façon dont la différence modulo est interprétée. (Par exemple, -2^{31} s .. + (2^{31}-1) s ou -2^k s .. + (2^{32}-2^k-1) s pour $0 \leq k < 32$, etc.)

NOTE 2 Les éléments suivants ne sont utilisés que dans des structures de données condensées.

```
Unsigned1::= INTEGER(0..1)                          -- 1-bit unsigned
Unsigned2::= INTEGER(0..3)                          -- 2-bit unsigned
Unsigned3::= INTEGER(0..7)                          -- 3-bit unsigned
Unsigned4::= INTEGER(0..15)                         -- 4-bit unsigned
Unsigned5::= INTEGER(0..31)                         -- 5-bit unsigned
Unsigned6::= INTEGER(0..63)                         -- 6-bit unsigned
Unsigned7::= INTEGER(0..127)                        -- 7-bit unsigned

Unsigned9::= INTEGER(0..511)                        -- 9-bit unsigned
Unsigned10:= INTEGER(0..1 023)                      -- 10-bit unsigned
Unsigned11::= INTEGER(0..2 047)                     -- 11-bit unsigned
Unsigned12::= INTEGER(0..4 095)                     -- 12-bit unsigned
Unsigned13::= INTEGER(0..8 191)                     -- 13-bit unsigned
Unsigned14::= INTEGER(0..16 383)                    -- 14-bit unsigned
Unsigned15::= INTEGER(0..32 767)                    -- 15-bit unsigned

Unsigned63::= INTEGER(0..9 223 372 036 854 775 807) -- 63-bit unsigned
```

12.23.1.4 En-tête de sous-couche d'application

```
RequestResponse::= Unsigned1 (
   request        (0),
   response       (1)
)

ObjectAddressingMode::= Unsigned2 (
   compact        (0)      -- indicates 4-bit object identifiers
   midSize        (1)      -- indicates 8-bit object identifiers
   fullSize       (2)      -- indicates 16-bit object identifiers
   inferred       (3)      -- shall only be used as specified in 12.22.2.4.5.
)

ASLService::= Unsigned5 (
   Publish                 0,
   AlertReport             1,
   AlertAcknowledge        2,
   Read                    3,
   Write                   4,
   Execute                 5,
   Tunnel                  6,
                           -- values 7..31 are reserved for future use by this standard
)
```

```
ASLConfirmedServiceRequest::= CHOICE(
   -- the first octet of the ConfirmedServiceRequest is constructed with
         -- bit 7 containing RequestResponse
         -- bits 6 and 5 containing ObjectAddressingMode
         -- bits 4..0 containing ASLService
   readCompact                    [3]   IMPLICIT ReadRequestPDU,      -- bit pattern: 0000 0011
   readMidSize                    [35]  IMPLICIT ReadRequestPDU,      -- bit pattern: 0010 0011
   readFull                       [67]  IMPLICIT ReadRequestPDU,      -- bit pattern: 0100 0011
   readInferred                   [99]  IMPLICIT ReadRequestPDU,      -- bit pattern: 0110 0011
   writeCompact                   [4]   IMPLICIT WriteRequestPDU,     -- bit pattern: 0000 0100
   writeMidSize                   [36]  IMPLICIT WriteRequestPDU,     -- bit pattern: 0010 0100
   writeFull                      [68]  IMPLICIT WriteRequestPDU,     -- bit pattern: 0100 0100
   writeInferred                  [100] IMPLICIT WriteRequestPDU,     -- bit pattern: 0110 0100
   executeCompact                 [5]   IMPLICIT ExecuteRequestPDU,   -- bit pattern: 0000 0101
   executeMidSize                 [37]  IMPLICIT ExecuteRequestPDU,   -- bit pattern: 0010 0101
   executeFull                    [69]  IMPLICIT ExecuteRequestPDU,   -- bit pattern: 0100 0101
   executeInferred                [101] IMPLICIT ExecuteRequestPDU,   -- bit pattern: 0110 0101
   tunnelCompact                  [6]   IMPLICIT TunnelRequestPDU,    -- bit pattern: 0000 0110
   tunnelMidSize                  [38]  IMPLICIT TunnelRequestPDU,    -- bit pattern: 0010 0110
   funnelFull                     [70]  IMPLICIT TunnelRequestPDU,    -- bit pattern: 0100 0110
   tunnelInferred                 [102] IMPLICIT TunnelRequestPDU     -- bit pattern: 0110 0110
)

ASLConfirmedServiceResponse::= CHOICE(
   -- the first octet of the ConfirmedServiceResponse is constructed with
         -- bit 7 (MSBO containing RequestResponse) = 1 -- only response form is valid
         -- bits 6 and 5 containing ObjectAddressingMode
         -- bits 4..0 containing ASLService
   readCompact                    [131]   IMPLICIT ReadResponsePDU, --bit pattern: 1000 0101
   readMidSize                    [163]   IMPLICIT ReadResponsePDU, --bit pattern: 1010 0011
   readFull                       [195]   IMPLICIT ReadResponsePDU, --bit pattern: 1100 0011
   readInferred                   [227]   IMPLICIT ReadResponsePDU, --bit pattern: 1110 0011
   writeCompact                   [132]   IMPLICIT WriteResponsePDU, --bit pattern: 1000 0100
   writeMidSize                   [164]   IMPLICIT WriteResponsePDU, --bit pattern: 1010 0100
   writeFull                      [196]   IMPLICIT WriteResponsePDU, --bit pattern: 1100 0100
   writeInferred                  [228]   IMPLICIT WriteResponsePDU, --bit pattern: 1110 0100
   executeCompact                 [133]   IMPLICIT ExecuteResponsePDU, --bit pattern: 1000 0101
   executeMidSize                 [165]   IMPLICIT ExecuteResponsePDU, --bit pattern: 1010 0101
   executeFull                    [197]   IMPLICIT ExecuteResponsePDU, --bit pattern: 1100 0101
   executeInferred                [229]   IMPLICIT ExecuteResponsePDU, --bit pattern: 1110 0101
   tunnelCompact                  [134]   IMPLICIT TunnelResponsePDU, --bit pattern: 1000 0110
   tunnelMidSize                  [166]   IMPLICIT TunnelResponsePDU, --bit pattern: 1010 0110
   funnelFull                     [198]   IMPLICIT TunnelResponsePDU, --bit pattern: 1100 0110
   tunnelInferred                 [230]   IMPLICIT TunnelResponsePDU --bit pattern: 1110 0110
)

ASLUnconfirmedServiceRequest::= CHOICE (
   -- the first octet of the UnconfirmedServiceRequest is constructed with
         -- bit 7 (MSBO containing RequestResponse) = 0 -- only request form is valid
         -- bits 6 and 5 containing ObjectAddressingMode
         -- bits 4..0 containing ASLService
   alertReportCompact             [1]   IMPLICIT AlertReportRequestPDU, --bit pattern: 0000 0001
   alertReportMidSize             [33]  IMPLICIT AlertReportRequestPDU, --bit pattern; 0010 0001
   alertReportFull                [65]  IMPLICIT AlertReportRequestPDU, --bit pattern: 0100 0001
   alertReportInferred            [97]  IMPLICIT AlertReportRequestPDU, --bit pattern: 0110 0001
   alertAcknowledgeCompact        [2]   IMPLICIT AlertAcknowledgeRequestPDU, --0x 0000 0010
   alertcknowledgeMidSize         [34]  IMPLICIT AlertAcknowledgeRequestPDU, --0x 0010 0010
   alertReportFull                [66]  IMPLICIT AlertAcknowledgeRequestPDU, --0x 0100 0010
   alertReportInferred            [98]  IMPLICIT AlertAcknowledgeRequestPDU, --0x 0110 0010
   tunnelCompact                  [6]   IMPLICIT TunnelRequestPDU, --bit pattern: 0000 0110
   tunnelMidSize                  [38]  IMPLICIT TunnelRequestPDU, --bit pattern: 0010 0110
   funnelFull                     [70]  IMPLICIT TunnelRequestPDU, --bit pattern: 0100 0110
   tunnelInferred                 [102] IMPLICIT TunnelRequestPDU --bit pattern: 0110 0110
)

ASLPublicationServiceRequest::= CHOICE (
   -- the first octet of the PublicationServiceRequest is constructed with
         -- bit 7 (MSBO containing RequestResponse) = 0 -- only request form is valid for publication)
         -- bits 6 and 5 containing ObjectAddressingMode
   publishCompact                 [0]   IMPLICIT PublishRequestPDU,   bit pattern: 0000 0000
   publishMidSize                 [32]  IMPLICIT PublishRequestPDU,   bit pattern: 0010 0000
   publishFull                    [64]  IMPLICIT PublishRequestPDU    bit pattern: 0100 0000
                  -- inferred addressing is not used as there is no concatenation of publications
                  -- (see concentrator / dispersion objects)
)
```

12.23.1.5 APDU individuelles

```
SourceAndDestinationOIDs:: = IMPLICIT SEQUENCE (OCTET ALIGNED)(
      IMPLICIT CHOICE ( -- as determined by objectAddressingMode in bits 5 and 6 of first octet of APDU
            -- source object represents the initiator of the service primitive (.req or .rsp)
            -- destination object represents the recipient of the service primitive (.ind or .cnf)
            compact IMPLICIT PACKED SEQUENCE (
                  compactSourceObject        Unsigned4,
                  compactDestinationObject   Unsigned4
            )
            midSize IMPLICIT SEQUENCE (
                  midSizeSourceOID           Unsigned8,
                  midSizeDestinationOID      Unsigned8
            )
            fullSize IMPLICIT SEQUENCE (
                  fullSizeSourceOID          Unsigned16,
                  fullSizeDestinationOID     Unsigned16
            )
            inferred NULL
      )

ReadRequestPDU::= IMPLICIT SEQUENCE
   soidDoid                                  SourceAndDestinationOIDs,
   readRequest                               ReadRequest
)

ReadResponsePDU::= IMPLICIT SEQUENCE
   soidDoid                                  SourceAndDestinationOIDs,
   readResponse                              ReadResponse
)

WriteRequestPDU::= IMPLICIT SEQUENCE
   soidDoid                                  SourceAndDestinationOIDs,
   writeRequest                              WriteRequest
)

WriteResponsetPDU::= IMPLICIT SEQUENCE
   soidDoid                                  SourceAndDestinationOIDs,
   writeResponse                             WriteResponse
)

ExecuteRequestPDU::= IMPLICIT SEQUENCE
   soidDoid                                  SourceAndDestinationOIDs,
   executeRequest                            ExecuteRequest
)

ExecuteResponsePDU::= IMPLICIT SEQUENCE
   soidDoid                                  SourceAndDestinationOIDs,
   executeResponse                           ExecuteResponse
)

TunnelRequestPDU::= IMPLICIT SEQUENCE
   soidDoid                                  SourceAndDestinationOIDs,
   tunnelRequest                             TunnelRequest
)

TunnelResponsePDU::= IMPLICIT SEQUENCE
   soidDoid                                  SourceAndDestinationOIDs,
   tunnelResponse                            TunnelResponse
)

AlertReportRequestPDU::= IMPLICIT SEQUENCE
   soidDoid                                  SourceAndDestinationOIDs,
   alertReportRequest                        AlertReportRequest
)

AlertAcknowledgeRequestPDU::= IMPLICIT SEQUENCE
   soidDoid                                  SourceAndDestinationOIDs,
   alertAcknowledgeRequest                   AlertAcknowledgeRequest
)
```

```
PublishRequestPDU::= IMPLICIT SEQUENCE
   soidDoid                         SourceAndDestinationOIDs,
   publishRequest                   PublishRequest
)
```

12.23.1.6 APDU périodiques

```
PublishRequest::= IMPLICIT SEQUENCE (
   IMPLICIT CHOICE ( -- implicitly determined by the corresponding application processes
      NativeValue                   IMPLICIT PublishedValue,              -- single published value
      NativeSequence                IMPLICIT PublishedValueSequence, -- sequence of published values
      HealthReportSequence          IMPLICIT HealthReportSequence,        -- publication HRCO
      nonNativeSequence             IMPLICIT NonNativeSequence            -- publication tunnel
       )
)

PublishedValue::= IMPLICIT SEQUENCE (
   contentVersion                   Unsigned8,    -- version of configuration of content published
   freshValueSequenceNumber         Unsigned8,    -- freshness of this set of data
   value                            ProcessValueAndStatus
)

PublishedValueSequence::= IMPLICIT SEQUENCE (
   contentVersion                   Unsigned8,    -- version of configuration of content published
   freshValueSequenceNumber         Unsigned8,    -- freshness of this set of data
   publishedValues                  SEQUENCE OF ProcessValueAndStatus
)

HealthReportSequence::= IMPLICIT SEQUENCE (
   contentVersion                   Unsigned8,    -- version of configuration of content published
   freshValueSequenceNumber         Unsigned8,    -- freshness of this set of data
   healthReportSize                 Unsigned8,
   healthReport                     OCTET STRING
)

NonNativeSequence::= IMPLICIT OCTET STRING

ProcessValueAndStatus::= IMPLICIT CHOICE ( -- based on publisher and subscriber application configuration
   analog         AnalogProcessValueAndStatus,
   boolean        BooleanProcessValueAndStatus
                  -- NOTE   This choice element can be extended by industry consortia and vendors
)

AnalogProcessValueAndStatus::= IMPLICIT SEQUENCE (
      valueStatus                   PV_Status,
      analogProcessValue            Float32
)

BooleanProcessValueAndStatus::= IMPLICIT SEQUENCE (
      valueStatus                   PV_Status,
      booleanProcessValue           Boolean8      -- single Boolean value represented by a full octet
)

PV_Status::= PACKED SEQUENCE (OCTET ALIGNED) ( -- 1 octet (bit field sizes are: 2 + 1 + 3 + 2)
   quality                          PV_Quality,              -- 2 bits
   reservedSpareBit                 Unsigned1,               -- 1 bit
   IMPLICIT CHOICE ( -- selected by quality; all are         -- 3 bits
      [0] BadValueSubstatus         BadValueSubstatus,
      [1] UncertainValueSubstatus   UncertainValueSubstatus,
      [2] GoodValueSubstatus        GoodValueSubstatus,
      [3] otherSubstatus            Unsigned3                          -- reserved for future use
    ),                                                                 -- 1 spare code point
   limitStatus                      LimitStatus              -- 2 bits control anti-windup information
)

PV_Quality::= Unsigned2 (                    -- 2 bits
   badValue,                        (0),                               -- value is bad
   uncertainValue                   (1),                               -- value is uncertain
   goodValue                        (2),                               -- value is good
   otherValue                       (3)                                -- reserved for future use
)                                                                      -- 1 spare code point
```

```
BadValueSubstatus::= Unsigned3 (                    -- 3 bits
   badValue_NonSpecific,                (0),
   badValue_ConfigurationError,         (1),
   badValue_NotConnected                (2),
   badValue_DeviceFailure,              (3),
   badValue_SensorFailure,              (4),
   badValue_NoCommunicationWithLUV      (5),
   badValue_NoCommunicationNoLUV        (6),
   badValue_OutOfService                (7)
)                                                   -- no spare code points

UncertainValueSubstatus::= Unsigned3 ( -- 3 bits
   uncertainValue_NonSpecific,          (0),
uncertainValue_LastUsableValue          (1),
uncertainValue_SubstitutedOrManualEntry (2),
uncertainValue_InitialValue             (3),
   uncertainValue_SensorConversionInaccurate, (4),
   uncertainValue_RangeLimitsExceeded   (5)
uncertainValue_SubNormal,               (6),
   uncertainValue_Spare                 (7)         -- reserved for future use
)                                                   -- 1 spare code point

GoodValueSubstatus::= Unsigned3 (       -- 3 bits
   goodValue_NoSpecialConditionsExist   (0),
   goodValue_SpecialCondition1          (1),        -- reserved for future use
   goodValue_SpecialCondition2          (2),        -- reserved for future use
   goodValue_SpecialCondition3          (3),        -- reserved for future use
   goodValue_SpecialCondition4          (4),        -- reserved for future use
   goodValue_SpecialCondition5          (5),        -- reserved for future use
   goodValue_SpecialCondition6          (6),        -- reserved for future use
   goodValue_SpecialCondition7          (7)         -- reserved for future use
)                                                   -- 7 spare code points

LimitStatus::= Unsigned2 ( -- 2 bits
   notLimited                           (0),
   lowLimited                           (1),
   highLimited                          (2),
   constant                             (3)    -- both high limited and low limited
)                                                   -- no spare code points

highLowLimited LimitStatus::= LimitStatus    constant        -- alternative symbolic name
lowHighLimited LimitStatus::= LimitStatus    constant        -- alternative symbolic name
```

12.23.1.7 APDU apériodiques

```
CompactObjectIdentifier::= Unsigned4
MidSizeObjectIdentifier::= Unsigned8
FullSizeObjectIdentifier::= Uinsigned16

ExtensibleInteger::= IMPLICIT SEQUENCE (OCTET ALIGNED) (
      format                  Boolean1, -- 1 bit, false for short form, true for long form
      IMPLICIT CHOICE ( -- choice is established by the format field
         shortForm            Unsigned7,      -- 7 bits     -- value shall be < 0x80
         longForm             Unsigned15,     -- 15 bits    -- value shall be  0ξ80 αvδ
       )                                      -- < 0x800; value < 0x80 are invalid
)
```

La taille minimale doit être utilisée pour le codage d'une donnée de type ExtensibleInteger. L'utilisation du format longForm pour une valeur qui pourrait être codée dans un format shortForm est invalide, et doit être rejetée comme une erreur de protocole.

```
AttributeClass::= Unsigned2 ( -- code points for attribute alternatives
      sixBitNoIndexing          (0),   -- 6-bit attribute identifier, no index
      sixBitOneDimension        (1),   -- 6-bit attribute identifier, one index (8 or 16 bits)
      sixBitTwoDimensions       (2),   -- 6-bit attribute identifier, two indices (each 8 or 16 bits)
      twelveBitExtended         (3)    -- 12-bit attribute identifier
)
```

```
TwelveBitIndexClass::= Unsigned2 ( – code points for 12-bit AID indexing alternatives
      twelveBitNoIndexing                 (0),
      twelveBitOneDimension               (1),
      twelveBitTwoDimensions              (2),
      twelveBitReserved                   (3)
)

ExtensibleAttributeIdentifier::= IMPLICIT PACKED SEQUENCE (OCTET ALIGNED) (
      attributeFormat                     AttributeClass          --2 bits
      IMPLICIT CHOICE ( -- choice is established by element attributeFormat
            sixBitNoIndexing              Unsigned6,
            sixBitOneDimension IMPLICIT SEQUENCE (OCTET ALIGNED) (
                  sixBitOneIndexAID       Unsigned6,
                  sixBitOneIndex          ExtensibleInteger,
            ),
            sixBitTwoDimensions IMPLICIT SEQUENCE (OCTET ALIGNED) (
                  sizBitTwoIndexAID       Unsigned6,
                  sixBitTwoIndexNo1       ExtensibleInteger,
                  sixBitTwoIndexNo2       ExtensibleInteger
            ),
            twelveBitExtended IMPLICIT SEQUENCE (OCTET ALIGNED) (
                  twelveBitIndex          TwelveBitIndexClass,
                  twelveBitAID            Unsigned12
                  CHOICE ( -- choice is established by the twelveBitIndexClass
                                          twelveBitNoIndexing NULL,
                                          twelveBitOneDimension: ExtensibleInteger,
                                          twelveBitTwoDimensions IMPLICIT SEQUENCE (OCTET ALIGNED) (
                                              TwelveBitTwoIndexNo1        ExtensibleInteger,
                                              TwelveBitTwoIndexNo2        ExtensibleInteger
                                          )
                  )
            )
      )
)
```

NOTE Les quatre bits dans le premier octet et les huit bits du deuxième octet attributeID sont concaténés pour former une plus longue valeur Unsigned12 quand l'alternative d'attributeID de douze bits est indiquée. Les quatre bits dans le premier octet sont de poids fort, et les huit bits dans le deuxième octet sont de poids faible.

```
ScalarType:::= Unsigned12 (
   Null                                (0)
   Boolean8                            (1),    -- single Boolean value represented by a full octet
   Integer8                            (2),
   Integer16                           (3),
   Integer32                           (4),
   Unsigned8                           (5).
   Unsigned16                          (6).
   Unsigned32                          (7),
   Float32                             (8),
   VisibleString                       (9),    -- GenericSizeAndValue format
   OctetString                         (10),   -- GenericSizeAndValue format


   BitString                           (14),

   Float64                             (30),
   TAITimeDifference                   (31),
   TAINetworkTime                      (32)
)                                          -- all other code points are reserved for this standard
```

Le codage de primitive doit être utilisé pour les éléments de valeur ScalarData, ArrayData, et StructureData. Aucune information de type n'est incluse dans le codage.

```
GenericSizeAndValue::= IMPLICIT SEQUENCE OF (
      SizeInOctets                        ExtensibleInteger,      -- necessary for parsing (e.g., concatenations)
      DataValue                           IMPLICIT SEQUENCE OF Octet1
)

ServiceFeedbackCodeGenericSizeAndValue::= GenericSizeAndValue
```

12.23.2 Rapports d'alerte et acquittements

```
AlertClass::= Unsigned1 (                                   -- 1 bit
   event                    (0),
   alarm                    (1)
)

AlertCategory::= Unsigned2 ( -- 2 bits
   deviceDiagnostic         (0),
   communicationsDiagnostic (1),
   security                 (2),
   process                  (3)
)

AlarmDirection::= Unsigned1 ( -- 1 bit
   returnToNormalOrNoAlarm  (0),   -- for alerts, set this value to 0; for alarm returns set this to zero
   inAlarm                  (1)    -- to report an alarm condition, set this value to 1.
)
```

La présente norme ne définit pas actuellement d'alertes normalisées pour les objets définis par l'AL et indépendants vis-à-vis de toute industrie ci-après:

– UAPMO;

– ARO;

– UDO;

– Concentrator;

– Dispersion;

– Tunnel;

– Interface.

```
ASLMO_Communication_Alerts::= ENUMERATED (
   malformed_APDU                (0),
                -- values 1..50 are reserved for future use by this standard
                -- values 51..100 are reserved for future use by standard profiles
                -- vendor-specific codes range 101..255
)

AI_ProcessAlerts::= ENUMERATED ( -- 1 octet;
   outOfServiceAlarm        (0),
   highAlarm                (1),
   highHighAlarm            (2),
   lowAlarm                 (3),
   lowLowAlarm              (4),
   deviationLowAlarm        (5),
   deviationHighAlarm       (6)
                -- values 7..50 are reserved for future use by this standard
                -- values 51..100 are reserved for future use by standard profiles
                -- vendor-specific codes range 101..255
)

AO_ProcessAlerts::= ENUMERATED ( -- 1 octet;
   outOfServiceAlarm        (0),
   highAlarm                (1),
   highHighAlarm            (2),
   lowAlarm                 (3),
   lowLowAlarm              (4),
   deviationLowAlarm        (5),
   deviationHighAlarm       (6)
                -- values 7..50 are reserved for future use by this standard
                -- values 51..100 are reserved for future use by standard profiles
                -- vendor-specific codes range 101..255
)
```

```
BI_ProcessAlerts::= ENUMERATED ( -- 1 octet;
   outOfServiceAlarm                    (0),
   discreteAlarm                        (1)
                -- values 2..50 are reserved for future use by this standard
                -- values 51..100 are reserved for future use by standard profiles
                -- vendor-specific codes range 101..255
)

BO_ProcessAlerts::= ENUMERATED (
   outOfServiceAlarm                    (0),
   discreteAlarm                        (1)
                -- values 2..50 are reserved for future use by this standard
                -- values 51..100 are reserved for future use by standard profiles
                -- vendor-specific codes range 101..255
 )

ARMO_Alerts::= ENUMERATED (
AlarmRecoveryStart                      (0),
AlarmRecoveryEnd                        (1)
                -- values 2..50 are reserved for future use by this standard
                -- values 51..100 are reserved for future use by standard profiles
                -- vendor-specific codes range 101..255
)

IndividualAlertID:: = Unsigned8                 -- unique ID associated with an individual alert
                                                -- assigned by the application process in the UAL

statusSignalNamur107::= Unsigned8 (
   failure                              (0),    --
   checkFunction                        (1),    --
   offSpec                              (2),    --
   maintenanceRequired                  (3),    --
)

IndividualAlert::= IMPLICIT PACKED SEQUENCE (OCTET ALIGNED)(
      individualAlertID                 IndividualAlertID,
      DetectingObjectTransportLayerPort Unsigned16,
      DetectingObject                   Unsigned16,
      DetectingObjectType               Unsigned16,
      detectionTimeTAINetworkTime,      -- 48 bits
      alertClass                        AlertClass,             -- 1 bit
      alarmDirection                    AlarmDirection,         -- 0: event or alarm return; 1: alarm report
      alertCategory                     AlertCategory,          -- 2 bits: device, comm, security, process
      alertPriority                     AlertPriority,          -- 4 bits
      alertType                         Unsigned8,              -- object category and type dependent
      associatedDataSize                ExtensibleInteger,
      associatedData                     -- present when associatedDataSize > 0
         CHOICE ( -- choice is based on AlertCategory
            communicationsDiagnostic IMPLICIT SEQUENCE OF Octet1 OPTIONAL,
            security IMPLICIT SEQUENCE OF Octet1 OPTIONAL,
            process IMPLICIT SEQUENCE OF Octet1 OPTIONAL,
            deviceDiagnostic IMPLICIT SEQUENCE
                  ( namur107Status          statusSignalNamur107,
                   detailedInformation      IMPLICIT SEQUENCE OF Octet1 OPTIONAL
                  )                             -- may include additional information per NAMUR-107
   ) OPTIONAL
)

AlertReportRequest::= ( -- note: client OID not present; ARMO is implied
      alert         IndividualAlert
)

AlertAcknowledgeRequest::= (
      alertID       IndividualAlertID                   -- server is always ARMO
)

AlertPriority::= Unsigned4
```

La priorité alerte est une valeur qui indique l'importance de l'alerte. Plus la valeur est grande, plus l'alerte est importante. Les systèmes d'hôte mettent en correspondance les priorités d'appareil aux priorités d'alerte d'hôte qui comprennent habituellement les catégories:

– urgent,

– high,
– medium,
– low, et
– journal.

Le mapping recommandé des valeurs de priorité d'alerte en ces catégories est spécifié en 12.17.5.2.2.22.

```
MalformedAPDUClass::= AlertClassevent;

MalformedAPDUAlertCategory::= AlertCategorycommunicationsDiagnostic

MalformedAPDUAlertType::= AlertTypemalformedAPDUCommunicationAlert

MalformedAPDUAlertPriority = 7          -- mid-range of medium priority alerts

MalformedPDUAlertValue::= IMPLICIT SEQUENCE (  -- alert value sent by ASL to DMAP
   sourceAddress                IPv6Address,          -- 128 bits to ensure address uniqueness.
   thresholdExceeded            Unsigned16,
   TimeWindow                   TAITimeDifference
)

MalformedPDUAlertValueSize::= 24 -- sizeof(MalformedPDUAlertValue)
```

12.23.3 Service feedback code

NOTE Service feedback code sert à indiquer le statut (succès, par exemple), un avertissement (valeur limitée, par exemple) ou une erreur (mode incompatible, par exemple).

```
ServiceFeedbackCode::= Unsigned8 (      -- 1octet
   -- standard error codes, range 0..127
   success                      (0)     -- success
   failure                      (1)     -- generic failure
   other                        (2),    -- reason other than that listed in this enumeration
   invalidArgument              (3),    -- invalid attribute to a service call
   invalidObjectID              (4),    -- invalid object ID
   invalidService  (5),                 -- unsupported or illegal service
   invalidAttribute             (6),    -- invalid attribute index
   invalidElementIndex          (7),    -- invalid array or structure element index (or indices)
   readOnlyAtrribute            (8),    -- read-only attribute
   valueOutOfRange              (9),    -- value is out of permitted range
   inappropriateProcessMode     (10),   -- process is in an inappropriate mode for the request
   incompatibleMode             (11),   -- value is not acceptable in current context

   invalidValue    (12),                -- value (data) not acceptable for other reason
                                           (e.g., too large, too small, invalid engineering units code)
   internalError    (13),               -- device internal problem
   invalidSize      (14),                  -- size is not valid (may be too big or too small)
   incompatibleAttribute        (15),   -- attribute not supported in this version
   invalidMethod  (16),                 -- invalid method identifier
   objectStateConflict          (17),   -- state of object in conflict with action requested
   inconsistentContent          (18),   -- the content of the service requested is inconsistent
   invalidParameter             (19),   -- value conveyed is not legal for method invocation
   objectAccessDenied           (20),   -- object is not permitting access
   typeMismatch                 (21),   -- data not as expected (e.g., too many or too few octets)
   deviceHardwareCondition      (22),   -- device specific hardware condition prevented request from
                                           succeeding (e.g., memory defect problem)
   deviceSensorCondition        (23),   -- problem with sensor detected
deviceSoftwareCondition         (24),   -- device specific software condition prevented request from
                                           succeeding (e.g., local lockout, local write protection,
                                        -- simulating in progress)
   fieldOperationCondition      (25),   -- field specific condition prevented request from succeeding
                                           (e.g., lockout, or environmental condition not in range)
   configurationMismatch        (26),   -- a configuration conflict was detected
   insufficientDeviceResources  (27),   -- e.g., queue full, buffers/memory unavailable
   valueLimited                 (28),   -- e.g., value limited by device
   dataWarning                  (29),   -- e.g., value has been modified due to a device specific reason
   invalidFunctionReference     (30),   -- function referenced for execution is invalid
   functionProcessError         (31),   -- function referenced could not be performed due to a device
                                           specific reason
```

```
   warning                          (32),   -- successful, but there is additional information that may be of
                                            interest to the user which may, for example be conveyed via
                                            accessing an attribute or by sending an alert
writeOnlyAttribute                  (33),   -- write-only attribute (e.g., a command attribute)
   operationAccepted                (34),   -- method operation accepted
   invalidBlockSize                 (35),   -- upload or download block size not valid
   invalidDownloadSize              (36),   -- total size for upload not valid
   unexpectedMethodSequence         (37),   -- required method sequencing has not been followed
   timingViolation                  (38),   -- object timing requirements have not been satisfied
   operationIncomplete              (39),   -- method operation, or method operation sequence not
                                            successful
   invalidData                      (40),   -- data received is not valid
                                            -- (e.g., checksum error, data content not as expected, etc.)

   dataSequenceError                (41),   -- data is ordered; data received is not in the order required
                                               example: duplicate data was received
   operationAborted                 (42),   -- operation aborted by server
   invalidBlockNumber               (43),   -- invalid block number
   blockDataError                   (44),   --error in block of data, example, wrong size, invalid content
   blockNotDownloaded               (45),   -- the specified block of data has not been successfully
                                            downloaded
   writeProtected                   (46),   -- data is write protected, so write operation is invalid
   invalidMode                      (47),   -- operation did not succeed due to invalid mode
      -- ...                               -- range 48..127 is reserved for future use of this standard
                                            -- vendor-specific device-specific feedback codes, range 128..255
   vendorDefinedError_128           (128),  -- redefinable by each device vendor for each device type
      -- ...
   vendorDefinedError_254           (254),  -- redefinable by each device vendor for each device type
   extensionCode                    (255)   -- indicates a two-octet field size for an extended service
                                                feedback code value
)                                           -- 123 values redefinable by each device vendor for each device type
```

12.23.4 Read, write et execute

```
RequestID::= Unsigned8
ReadRequest::= IMPLICIT SEQUENCE (
   requestID                        RequestID,
   targetAttribute                  ExtensibleAttributeIdentifier
)

ApduResponseControlData::= PACKED IMPLICIT SEQUENCE (
   ,
   Spare                            Unsigned7,     -- redefinable in future editions of this standard
   ForwardCongestionNotificationEcho Boolean1  -- true when congestion in forward (request) path detected
)

ReadResponse::= IMPLICIT SEQUENCE (
   requestID                        RequestID,              -- matches corresponding ReadRequest
   apduControl                      ApduResponseControlData,
   readValue                        ServiceFeedbackCodeGenericSizeAndValue
)

WriteRequest::= IMPLICIT SEQUENCE (
   requestID                        RequestID,
   targetAttribute                  ExtensibleAttributeIdentifier
   value                            GenericSizeAndValue
)

WriteResponse::= IMPLICIT SEQUENCE (
   requestID                        RequestID,              -- matches corresponding WriteRequest
   apduControl                      ApduResponseControlData,
   serviceFeedbackCode              ServiceFeedbackCode
)

MethodInvocationRequest::= IMPLICIT SEQUENCE (
   methodID                         Unsigned8,
   requestParametersSize            ExtensibleInteger,
   requestParameters                IMPLICIT SEQUENCE of Octet1 OPTIONAL
      -- primitive encoding; data type known by correspondents
      -- requestParameters only present if requestParametersSize >0
)
```

```
MethodInvocationResponse::= IMPLICIT SEQUENCE (
   responseParametersSize              ExtensibleInteger,
   responseParameters                  IMPLICIT SEQUENCE of Octet1 OPTIONAL
      -- primitive encoding; data type known by correspondents
      -- responseParameters only present if responseParametersSize >0
)

ExecuteRequest:: = IMPLICIT SEQUENCE (
      requestID                        RequestID,
      methodinvocationRequest          MethodInvocationRequest -- data type(s) specified by standard
)

ExecuteResponse::= IMPLICIT SEQUENCE (
      requestID                        RequestID,
      apduControl                      ApduResponseControlData,
      serviceFeedbackCode              ServiceFeedbackCode,
      methodInvocationResponse         MethodInvocationResponse -- data type(s) specified by standard
)
```

12.23.5 Tunnel

```
TunnelRequest::= IMPLICIT SEQUENCE (
   length                              ExtensibleInteger,
   tunnelPayload                       SEQUENCE OF Octet1
)

TunnelResponse::= IMPLICIT SEQUENCE (
   apduControl                         ApduResponseControlData,
   length                              ExtensibleInteger,
   tunnelPayload                       SEQUENCE OF Octet1
)
```

12.23.6 Fin de module contenu

```
FIN
```

12.24 Exemples de codage détaillés (informative)

12.24.1 Read

Scénario: L'objet client 11 souhaite lire des données dans l'objet serveur 12, attribut 3. La réponse indique que la lecture a réussi et retourne une valeur de longueur deux octets.

Le Tableau 364 montre un exemple d'une demande de lire plusieurs valeurs.

Tableau 364 – Exemple de codage: Demande Read pour un attribut sans indice

Codage des octets en hexadécimal	Sémantique
03	Demande de lecture
BC	Client (source) object ID = 11_{10} Server (destination) object ID = 12_{10}
XX	Identificateur de demande
03	ID d'attribut = 3 (l'attribut est scalaire)

Le Tableau 365 montre un exemple d'une réponse à une demande de lire plusieurs valeurs.

Tableau 365 – Exemple de codage: Réponse Read pour un attribut sans indice

Codage des octets en hexadécimal	Sémantique
83	Réponse Read
CB	Server (source) object iD = 12_{10} Client (destination) object ID = 11_{10}
XX	Identificateur Request (même valeur que pour l'identificateur Request qui était inclus dans la demande de service correspondante)
00	Succès
02	La valeur a une longueur de deux octets
YY YY	Valeur

12.24.2 Tunnel

Scénario: L'objet 16 dans le client envoie un message à l'objet 20 dans le serveur. Le contenu du message doit être passé à l'objet serveur.

Le Tableau 366 montre un exemple d'une demande de service Tunnel qui a la taille de charge utile de 9 octets.

Tableau 366 – Exemple de codage: Demande de service Tunnel

Codage des octets en hexadécimal	Sémantique
06	Demande de Tunnel
09	Taille
(9 octets de données tunnellisées)	Données tunnellisées

13 Configuration

13.1 Généralités

Un appareil conforme à la présente norme est considéré comme étant configuré lorsque l'appareil a les informations exigées pour communiquer avec un réseau cible et pour initier une demande de rattachement au gestionnaire de système/sécurité du réseau cible. Dans le présent document, un réseau cible est défini comme étant un réseau que l'appareil est configuré pour rejoindre. Les informations exigées pour initier la demande de rattachement comprennent à la fois les informations (relatives à la confiance) de sécurité et les informations relatives à un réseau. L'Article 13 spécifie la procédure de configuration par liaison radio et le format de message dans lequel le moyen de terrain de Type A est utilisé, et les formats de message hors bande dans lesquels le moyen de terrain de Type A n'est pas utilisé pour configurer les informations relatives à la sécurité et relatives à un réseau.

La configuration par liaison radio utilise le processus de rattachement de sous-réseau pour établir une connexion entre l'appareil de configuration et l'appareil configuré. Le processus de rattachement de sous-réseau est décrit en 6.3.9.2 et suit deux chemins facultatifs, l'un défini pour un appareil se rattachant avec des informations relatives à la confiance basées sur une clé symétrique, et l'autre défini pour un appareil se joignant avec des informations relatives à la confiance basées sur une clé asymétrique. La configuration hors bande peut ne pas utiliser le processus de rattachement de sous-réseau; à la place, il peut configurer les informations par l'intermédiaire d'autres moyens câbles ou sans fil.

Le but du processus de configuration est de fournir assez d'informations afin que l'un de ces chemins puisse être pris par l'appareil.

Le processus de configuration implique un appareil qui met en œuvre le rôle de configuration en fournissant les informations relatives à un réseau et les informations relatives à la confiance au nouvel appareil. Pendant la configuration, l'opérateur peut utiliser l'appareil de configuration, agissant comme un proxy pour le gestionnaire de système, pour décider si, oui ou non, il convient de connecter un nouvel appareil au réseau, avec des informations issues du gestionnaire de sécurité. Dans un autre exemple, une copie de la liste d'appareils autorisés peut être obtenue auprès du gestionnaire de sécurité, permettant à l'appareil de configuration de prendre une décision locale. Lorsque le réseau cible est un réseau sécurisé, il est nécessaire de configurer des informations tant relatives à la confiance que relatives à un réseau; pour les réseaux non sécurisés, la clé par défaut (K_global) est utilisée comme information relative à la confiance. Une fois qu'un appareil est configuré, il est prêt à rejoindre le réseau cible. Par la suite, habituellement sans intervention humaine, le gestionnaire de sécurité du réseau cible peut soit accepter, soit rejeter, la demande de rattachement au réseau cible issue de l'appareil en se basant sur les informations configurées.

NOTE Dans la présente norme, divers aspects relatifs à l'installation d'informations relatives à la confiance et relatives à un réseau dans un appareil, l'acheminement de ces informations vers le gestionnaire de sécurité, et l'établissement de la confiance sont décrits dans différents articles. L'installation d'informations relatives à la confiance et relatives à un réseau est décrite à l'Article 13. L'acheminement d'informations vers le gestionnaire de sécurité est décrit à l'Article 9 et à l'Article 10. L'établissement de la confiance est décrit en 7.4.4.3.2.

13.2 Termes et définitions pour les appareils ayant divers rôles ou états

Les termes suivants sont définis pour les appareils ayant divers rôles ou états:

- **Appareil configuré**: Un appareil qui a besoin d'être configuré ou qui est en train d'être configuré. L'appareil peut manquer de tout ou partie des informations exigées pour rejoindre un réseau.

NOTE 1 Un appareil qui contient de vieilles informations relatives à un réseau est souvent mis à jour en le configurant avec de nouvelles informations.

- **Réseau cible**: Le réseau auquel le DBP est configuré pour se rattacher.
- **Appareil de configuration**: Un appareil qui met en œuvre le rôle consistant à configurer un autre appareil pour permettre à ce dernier appareil de rejoindre le réseau cible. Un PD peut ne pas être un appareil mettant en œuvre seulement le rôle de configuration; à la place, il pourrait être:
  - le gestionnaire de système/gestionnaire de sécurité du réseau cible;

    NOTE 2 Le rôle de gestionnaire de système/gestionnaire de sécurité peut être réparti, par exemple, dans un ensemble désigné d'appareils dans le réseau cible.

  - un appareil, tel qu'un appareil tenu à la main contenant un gestionnaire de système/gestionnaire de sécurité, qui utilise la suite de protocoles spécifiée par la présente norme pour configurer le DBP à travers un mini-réseau temporaire distinct; ou
  - un appareil qui utilise la communication hors bande (OOB), telle que communications en champ proche de champ (NFC) dans l'infrarouge, ou des prises, pour configurer le DBP, où la communication OOB ne relève pas du domaine d'application de la présente norme.
- **Réseau par défaut**: Le réseau dont l'identificateur de réseau est 1.
- **Réseau de configuration**: Un réseau formé entre le PD et le DBP. Si le PD est un appareil tenu à la main, le mini-réseau de configuration est le réseau formé entre l'appareil tenu à la main et le DBP. Si l'appareil de configuration est le gestionnaire de sécurité du réseau cible, le réseau de configuration peut être un réseau logique distinct séparé sur le réseau cible lui-même.
- **Clé de rattachement (K_join)**: Une clé de rattachement symétrique utilisée pour rejoindre un réseau cible sécurisé. La valeur de la clé K_join est censée être secrète, et elle est donc censée offrir la confidentialité de données. La valeur de la clé K_join

est mise à jour pendant la configuration à une nouvelle valeur qui est connue seulement par le gestionnaire de sécurité de réseau-cible et à l'appareil.[12]

- **Valeur de la clé de rattachement par défaut (K_global)**: Une clé de rattachement symétrique avec une valeur éditée. La valeur de K_global n'est pas censée être un secret; sa valeur est bien connue. Elle n'offre donc pas la confidentialité des données, mais elle aide à améliorer l'intégrité des données. Son but est d'établir la connectivité entre les appareils conformes à la présente norme qui ne partagent pas une clé de rattachement secrète. Une telle connectivité est nécessaire pour:
  - la configuration par liaison radio (OTA) des informations relatives à un réseau cible;
  - la lecture par OTA de l'identité de l'appareil et les réglages de configuration;
  - l'authentification par OTA des authentifiants de l'appareil; et
  - la mise à jour par OTA de la clé de rattachement K_join (les deux dernières étapes en utilisant la cryptographie asymétrique).

  La valeur de la clé de rattachement par défaut doit être K_global, définie en 7.2.2.2.
- **Clé de rattachement ouverte (K_join = K_open)**: Une valeur éditée non secrète pour la clé de rattachement (K_join). Cette valeur spéciale pour la clé de rattachement est utilisée pour rejoindre un réseau de configuration afin que certaines méthodes de configuration par OTA à clé symétrique seulement puissent être facilitées. La valeur réelle pour cette clé est définie en 7.2.2.2.
- **Réseaux physiques et logiques**: Un réseau physique est un ensemble d'appareils physiques qui communiquent les uns avec les autres, possiblement en plusieurs sauts. Un réseau logique est une instance de réseau qui fonctionne sur le réseau physique. Un réseau physique peut prendre en charge plusieurs réseaux logiques. Les réseaux logiques ont différentes propriétés individuelles de priorité et de sécurité. Par exemple, le réseau cible et le réseau de configuration sont deux réseaux logiques qui existent sur un réseau physique.
- **Etat inactif**: État d'appareil qui ne participe pas activement au réseau sans fil.
- **Etat de configuration**: L'appareil est dans la phase de configuration.
- **Etat "provisioned"**: L'appareil a reçu assez d'informations pour rejoindre le réseau cible, et a obtenu le DMO.Join_Command=1.
- **Valeur d'usine par défaut**: La configuration par défaut d'un appareil de terrain tel qu'il sort d'une installation de fabrication. La configuration par défaut a les valeurs de K_global et K_join égales à leurs valeurs par défaut, et la configuration par OTA permise. Un appareil opérationnel peut être réinitialisé à la valeur d'usine par défaut, soit par le gestionnaire de système lorsqu'il est une partie intégrante d'un réseau sécurisé, soit par un moyen OOB utilisant un appareil de configuration. Les valeurs d'usine par défaut pour le processus de configuration sont résumées dans le Tableau 367. Seul le gestionnaire de système doit avoir l'autorité de réinitialiser un appareil aux valeurs d'usine par défaut par l'intermédiaire du réseau.

La présente spécification n'exclut pas les appareils qui n'autorisent pas la réinitialisation aux valeurs d'usine par défaut.

---

[12] Des mécanismes appropriés sont fournis afin que la suite de protocoles définie par la présente norme ne puisse pas être utilisée pour lire dans un appareil la valeur courante de la clé de rattachement. Noter que le caractère secret des clés de rattachement ne peut pas être mis en vigueur par la présente norme.

13.3 Procédures de configuration

Tous les appareils de terrain conformes à la présente norme doivent mettre en œuvre un objet normalisé appelé objet de configuration d'appareil (DPO). Les attributs du DPO dans le DBP doivent spécifier les informations exigées pour initier une demande de rattachement au réseau cible. L'appareil doit retenir tous les attributs du DPO en cas de cycle d'alimentation en énergie ou de remplacement de batterie. L'objet de configuration d'appareil est décrit dans le détail en 13.9.1.

La présente spécification n'exclut pas que le gestionnaire de système puisse avoir le DPO, par exemple, qui stocke une adresse EUI64 du gestionnaire de sécurité.

Les PD doivent mettre en œuvre un objet de service de configuration d'appareil (DPSO) qui contient des informations destinées aux DBP qui sont desservis par le PD.

La configuration implique de positionner les attributs du DPO. Les attributs dans le DPO contiennent des informations tant relatives à un réseau que relatives à la confiance. Ces attributs peuvent être positionnés par l'intermédiaire de trois moyens différents:

- ils peuvent être préinstallés au cours de la fabrication de l'appareil; ou
- ils peuvent être positionnés en utilisant des moyens OOB; ou
- ils peuvent être positionnés par un PD utilisant un réseau de configuration, où le PD agit comme un proxy pour un gestionnaire de sécurité/gestionnaire de système d'un réseau cible pour fournir les informations relatives à la confiance et relatives à un réseau pour le réseau cible en question.

Tous les appareils conformes à la présente norme doivent prendre en charge la formation du réseau de configuration en utilisant seulement la suite complète de protocoles définie par la présente norme (PhL, DL, NL, et TL), c'est-à-dire, n'exigeant aucun autre mécanisme. Cependant, la présente norme n'interdit pas la configuration par des moyens de communication OOB.

Avec le moyen de terrain de Type A (5.2.6.4) dans le réseau de configuration, les PDU normalisées doivent être utilisées pour positionner les attributs du DPO, qui définit un ensemble d'attributs en lecture seule par défaut pour la formation du réseau de configuration ou d'un réseau non sécurisé. Les attributs par défaut incluent des clés symétriques par défaut éditées (K_join = K_global et K_join = K_open), un identificateur de sous-réseau D par défaut et un ensemble de voies par défaut. Comme cet ensemble d'attributs par défaut est connu et contenu dans le DPO de tous les appareils conformément à la présente norme, ces attributs fournissent un moyen pour tous les appareils de rejoindre un réseau de configuration.

Le DPO inclut un attribut, DPO.Allow_Provisioning, qui spécifie si l'accès aux attributs de l'objet par l'intermédiaire de l'instance ouverte par défaut est autorisé ou bloqué.

Quelques appareils peuvent mettre en œuvre un mécanisme externe (à savoir un commutateur) qui verrouillera l'état de configuration (soit vide, soit configuré) de l'appareil, pour réduire au maximum la consommation de batterie et pour réduire au maximum également la probabilité qu'un PD larron reconfigure un appareil.

13.4 Clés symétriques préinstallées

La formation d'un réseau de configuration n'est pas une étape nécessaire pour la configuration; les informations relatives à la confiance peuvent être préinstallées dans un appareil. Par exemple, il est possible qu'un utilisateur délègue (en partie) la configuration d'appareils à un fabricant d'appareil ou à un tiers. Un fabricant d'appareil peut préprogrammer des clés de rattachement symétriques secrètes et il peut fournir les données de clés de rattachement symétriques secrètes à l'utilisateur afin que les données puissent être chargées dans le gestionnaire de système/gestionnaire de sécurité du réseau cible. En variante, l'utilisateur peut stipuler au fabricant d'appareil la clé symétrique qui doit être chargée. Dans

ce cas, le DPO d'un appareil doit être préinstallé avec des informations relatives à un réseau cible et avec la clé de rattachement symétrique cible K_join. En fonction de l'application, deux ou plus de deux appareils peuvent partager les mêmes informations secrètes. Les appareils avec préinstallation d'informations relatives à la confiance et d'informations relatives à un réseau-cible peuvent passer directement au processus de rattachement de sous-réseau.

Lorsqu'un appareil a des informations relatives à la confiance qui sont préinstallées, mais pas d'informations relatives à un réseau-cible, il doit être possible de configurer l'appareil avec les informations nécessaires relatives au réseau. Cela facilite le fait d'avoir l'appareil qui reçoit des annonces issues du réseau-cible sur les voies prévues, en accélérant le processus de rattachement et les demandes de rattachement de sous-réseau présentes seulement vers le réseau-cible. Si les informations relatives à un réseau ne sont pas configurées, un appareil peut utiliser les réglages de réseau par défaut pour rechercher par balayage les annonces issues de tous les réseaux dans son voisinage (y compris ceux des concurrents de l'organisation propriétaire de l'appareil).

13.5 Configuration utilisant des mécanismes hors bande

Des appareils sans clés symétriques préinstallées ont besoin d'être authentifiés et ensuite configurés avec des informations de confiance. Comme noté plus haut, cela peut être accompli soit par le moyen de terrain de Type A de réseau de configuration par liaison radio, soit pas des mécanismes OOB.

Les moyens de communication OOB comprennent, sans s'y limiter, les connecteurs câblés en infrarouge, les cartes mémoire, les claviers sur des appareils, les NFC, et les prises. Le mécanisme des communications OOB ne relève pas du domaine d'application de la présente norme. Il convient que les attributs du DPO qui spécifient le rattachement au réseau-cible soient mis aux mêmes valeurs quel que soit le moyen utilisé (liaison radio ou OOB).

13.6 Réseaux de configuration

13.6.1 Généralités

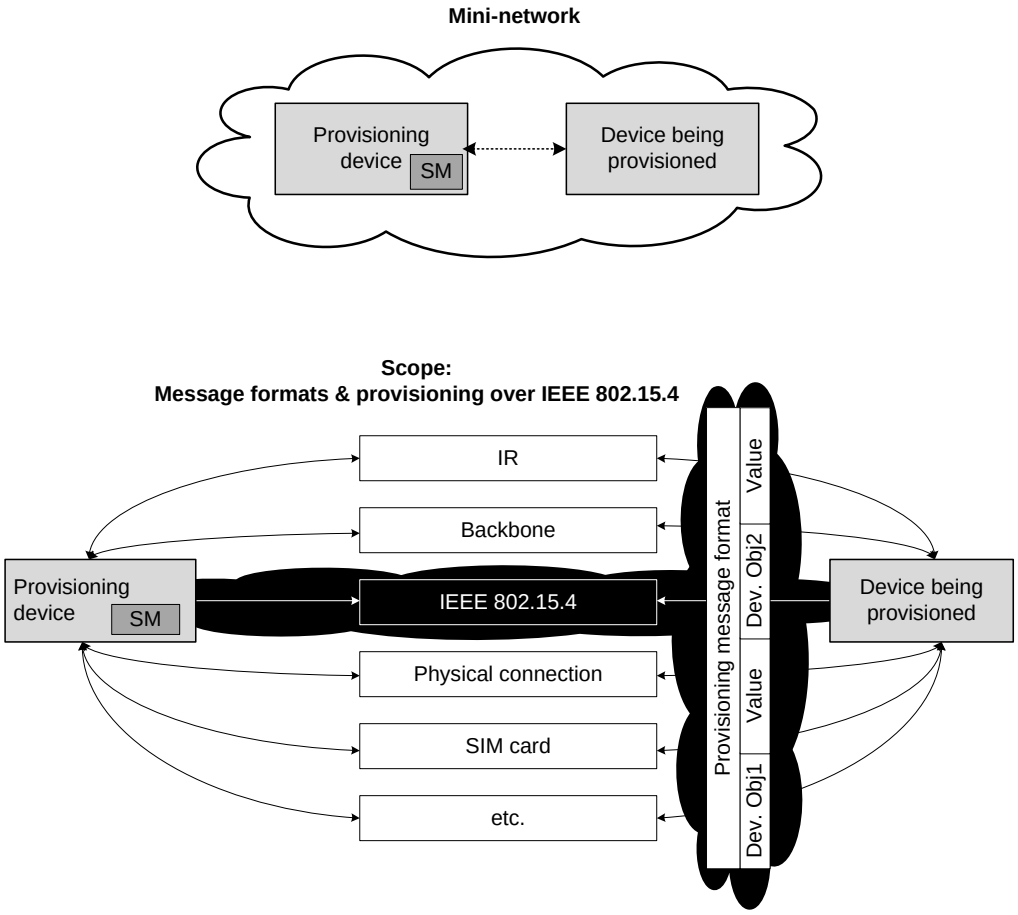
Outre la configuration OOB et la préconfiguration en usine, la présente norme définit la formation d'un réseau normalisé pour configurer des appareils par liaison radio (OTA) en utilisant le moyen de terrain de Type A. La clé de rattachement symétrique par défaut (K_global) ou la clé de rattachement symétrique ouverte (K_join = K_open) peut être utilisée comme étant les informations relatives à la confiance pour la formation du réseau de configuration par OTA. La clé de rattachement par défaut (K_global) est utilisée pour la formation du réseau de configuration pour obtenir des informations relatives à un réseau cible et la clé de rattachement à un réseau cible et pour des appareils avec une capacité cryptographique asymétrique. La clé de rattachement (K_join = K_open) est utilisée pour la formation d'un réseau de configuration où les informations relatives à la confiance et relatives à un réseau sont configurées par liaison radio. Cette forme de configuration est peu sûre et, par défaut, les gestionnaires de système et les appareils de configuration ne doivent pas permettre le rattachement avec cette clé de rattachement.

Un PD qui a des capacités cryptographiques asymétriques distingue la méthode avec la clé utilisée pour générer le MIC dans la primitive Security_Sym_Join().request. Dans le PD, le MIC généré par l'appareil rejoignant le réseau par défaut a besoin d'être validé un maximum de deux fois – une fois pour K_open et l'autre pour K_global. Si le gestionnaire de sécurité détecte que K_global est utilisée pour le MIC, le DBP doit être configuré en utilisant la cryptographie asymétrique. Autrement, le DBP doit être configuré en utilisant la clé symétrique K_open.

Le réseau de configuration peut soit être un mini-réseau isolé formé avec un appareil tenu à la main, soit être un réseau logique distinct sur le réseau cible lui-même. Dans le dernier cas, la connectivité allant du DBP au routeur d'annonce est ouverte, mais la connectivité allant plus loin, de ce routeur d'annonce vers le gestionnaire de système, est protégée par une session existante et elle est donc sécurisée. Si le réseau de configuration logique est sur le

réseau cible, les objets d'application des gestionnaires de système/de sécurité sur le réseau cible et le réseau de configuration logique (par exemple, DPSO) peuvent communiquer les uns avec les autres au sein du même appareil.

La Figure 135 montre le (mini-)réseau de configuration.



Légende

Anglais	Français
Mini-network	Mini-réseau
Provisioning device	Appareil de configuration
Device being provisioned	Appareil configuré
Scope: Message formats & provisioning over IEEE 802.15.4	Domaine: formats de message et configuration sur IEEE 802.15.4
Backbone	Dorsale
Provisioning device	Appareil de configuration
Physical connection	Connexion physique
SIM card	Carte SIM
Provisioning message format	Format de message de configuration
Dev. Obj1	Objet appareil 1
Value	Valeur
Dev. Obj2	Objet appareil 2
Device being provisioned	Appareil en cours de configuration

Figure 135 – Le réseau de configuration

La configuration par OTA utilise un PD qui peut être soit:

- un configurateur tenu à la main qui forme un mini-réseau isolé avec le DBP. Cet appareil tenu à la main a son propre gestionnaire de système/gestionnaire de sécurité et une fonctionnalité de routeur d'annonce; soit
- le gestionnaire de système/gestionnaire de sécurité du réseau de configuration logique sur le réseau cible.

NOTE Lorsqu'un PD est utilisé pour la configuration par OTA, il forme temporairement un mini-réseau et fonctionne comme étant à la fois le gestionnaire de système et le gestionnaire de sécurité pour le DBP.

13.6.2 Configuration par liaison radio utilisant la cryptographie asymétrique

Les DBP qui sont capables d'accomplir des calculs cryptographiques asymétriques doivent utiliser la clé de rattachement par défaut (K_global) pour rejoindre un réseau de configuration. Le DBP reçoit des annonces dont l'ID de sous-réseau D = 1 de routeurs d'annonce limitrophes et initie une demande de rattachement en utilisant la clé symétrique par défaut. Un processus de rattachement de sous-réseau réussi se traduit par le fait que le PD et le DBP ont établi un contrat pour une communication plus approfondie. Le PD utilise alors les primitives d'AL normalisées (telles que read et write) pour transférer les informations relatives à un réseau contenues dans son DPSO vers le DPO du DBP.

Pour configurer les informations relatives à la confiance, le PD interroge le DBP, à savoir., il lit ses authentifiants (par exemple: DPO.PKI_Certificate ou plusieurs DPO.PKI_Certificates, voir Annexe G), et envoie ces authentifiants vers le gestionnaire de sécurité. Le gestionnaire de sécurité du réseau de configuration vérifie les authentifiants du DBP et valide l'authenticité du DBP par un mécanisme de défi-réponse. Le gestionnaire de sécurité/gestionnaire de système peut demander une confirmation complémentaire de l'utilisateur par une GUI pour configurer le DBP. Une fois accepté, le gestionnaire de système fournit au DBP la clé de rattachement secrète, K_join, afin que le DBP puisse rejoindre le réseau cible soit immédiatement, soit plus tard, en utilisant la clé de rattachement en question. Lorsque cette nouvelle clé est émise par liaison radio, elle doit être chiffrée par la clé asymétrique du DBP, qui est une partie intégrante de son certificat, afin qu'elle ne puisse pas être récupérée par un auditeur frauduleux pendant qu'elle est en transit.

Des gestionnaires de sécurité conformes à la présente norme ne sont pas tenus d'avoir des capacités cryptographiques asymétriques; par conséquent, certains gestionnaires de sécurité peuvent ne pas être capables d'accepter ou de configurer des appareils en utilisant des capacités cryptographiques asymétriques. Lorsque le DBP rejoint le réseau de configuration en utilisant K_global, les gestionnaires de sécurité et les PD incapables d'accomplir des calculs cryptographiques asymétriques ne doivent pas émettre les informations relatives à la confiance vers le DBP.

En plus d'un niveau élevé de sécurité, les modules cryptographiques asymétriques et les certificats fournissent un moyen commode facile pour que les appareils établissent la communication avec le gestionnaire de sécurité du réseau-cible et soient configurés sans utilisation d'outils supplémentaires. Il est recommandé que les fabricants des gestionnaires de sécurité qui ne prennent pas en charge la cryptographie asymétrique fournissent des moyens adéquats (par exemple: mémoire, puissance de traitement ou périphériques complémentaires, etc.) pour mettre à niveau de tels gestionnaires de sécurité, à la demande d'utilisateur, pour prendre en charge la cryptographie asymétrique.

13.6.3 Configuration par liaison radio utilisant une clé de rattachement symétrique ouverte

La présente norme permet à des PD de configurer des appareils qui n'ont pas de capacités cryptographiques asymétriques d'être configurés par liaison radio. À cette fin, une clé de rattachement symétrique ouverte bien connue (K_join = K_open) est utilisée. Par défaut, le gestionnaire de sécurité dans le PD ne doit pas permettre la configuration par OTA avec la clé symétrique ouverte, K_open. Le réseau de configuration n'est pas sécurisé de lui-même, car il utilise une clé ouverte éditée et une clé de rattachement pour le réseau cible, et exige

donc des mesures de compensation, telles qu'une connexion physique sécurisée ou l'utilisation de la cryptographie asymétrique, pour la sécurité.

NOTE 1 Dans la configuration par OTA avec K_global, les informations de sécurité (à savoir la clé de rattachement) sont chiffrées avec une clé asymétrique pendant l'émission.

NOTE 2 L'utilisation d'une clé de rattachement symétrique ouverte pour la configuration n'est pas une procédure sécurisée. Un appareil d'écoute frauduleuse peut être capable d'obtenir les clés de rattachement au réseau cible et poser un risque de sécurité lorsque cette procédure de configuration est utilisée. Cette procédure de configuration peut seulement être utilisée dans des applications où le risque de sécurité est minimal et l'utilisateur n'est pas concerné ou a pris des mesures suffisantes pour éviter l'écoute frauduleuse. Une telle exposition peut être évitée en utilisant la configuration à chiffrement asymétrique ou la configuration OOB.

L'utilisation de la clé de rattachement symétrique K_open pour la configuration est une option de configuration. Les DBP peuvent être préconfigurés pour ne pas utiliser la configuration par OTA avec cette clé. Par défaut, les gestionnaires de sécurité et le PDS doivent rejeter les demandes de rattachement issues de tous les appareils qui envoient des demandes de rattachement en utilisant la clé de rattachement symétrique K_open. Les gestionnaires de sécurité et les PD ont besoin d'être configurés pour accepter des demandes de rattachement utilisant la clé de rattachement K_open. Il est admissible que des gestionnaires de sécurité et des PDS ne permettent pas une telle configuration.

Un appareil qui rejoint un réseau de configuration en utilisant la clé de rattachement K_open peut être configuré par le PD avec la clé de rattachement pour un réseau cible. Cependant, une fois configuré avec une nouvelle clé de rattachement pour le réseau cible, l'appareil ne doit être pas autorisé à utiliser la clé de rattachement symétrique K_open, à moins que l'appareil ne soit réinitialisé aux valeurs d'usine par défaut. Donc, le seul moyen admissible pour que l'appareil réutilise cette clé pour la configuration est de réinitialiser l'appareil aux valeurs d'usine par défaut.

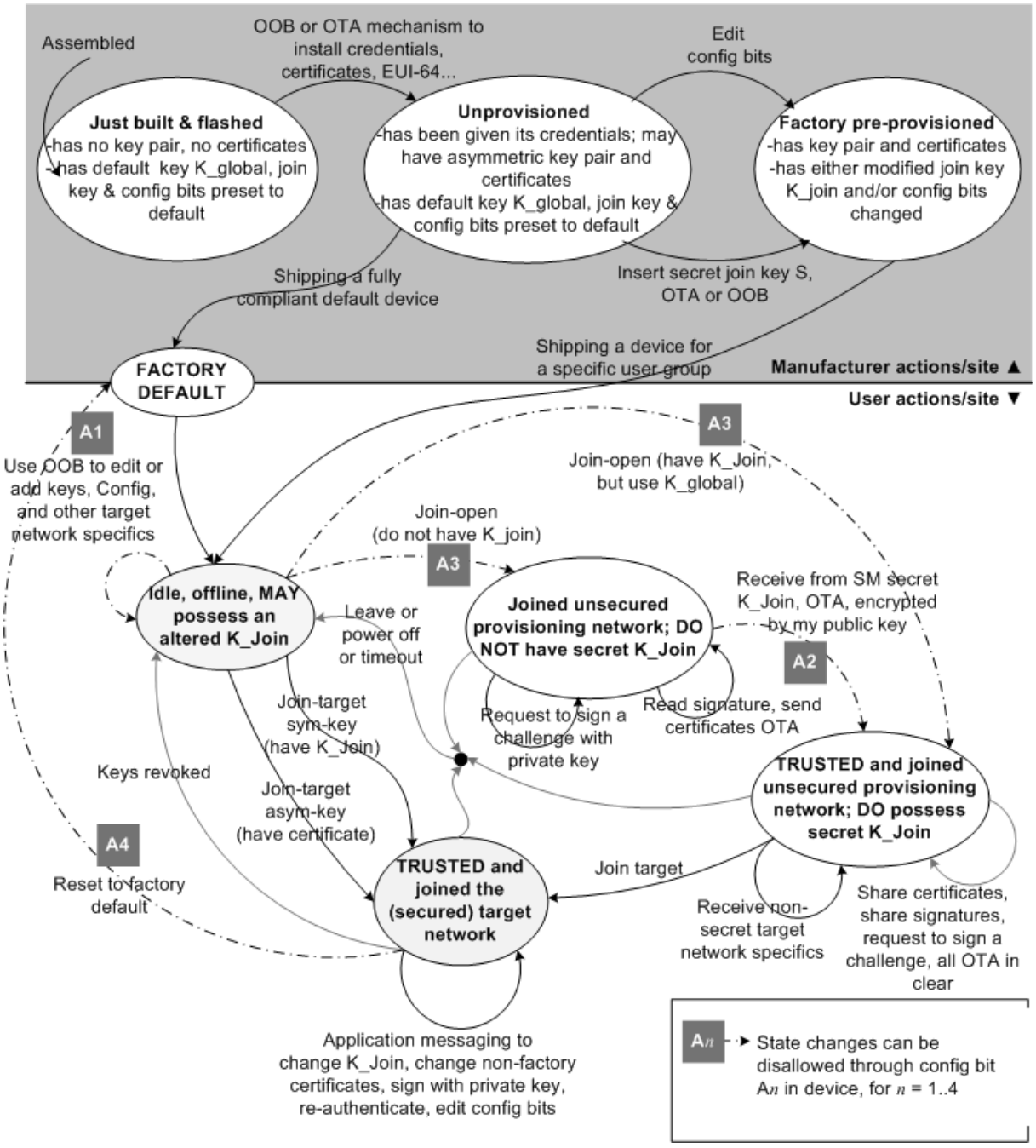
Le procédure de configuration utilisant la clé de rattachement symétrique K_open peut être utilisée dans le (mini-)réseau de configuration ou par l'intermédiaire d'un réseau logique distinct sur le réseau cible. Le DBP reçoit une annonce issue d'un réseau de configuration et une demande de rattachement normalisée est envoyée en utilisant une clé symétrique (K_join = K_open). Si la demande est acceptée, le DBP rejoint le réseau de configuration et un contrat est établi entre le DBP et le PD. Les primitives et les méthodes read/write à un niveau application sont disponibles au PD pour configurer les informations relatives à la confiance et relatives à un réseau; cela inclut, par exemple, de configurer la clé de rattachement à la cible en utilisant la méthode de DPO.Write_Join_Key.

Le (mini-)réseau de configuration peut également être utilisé pour la configuration d'appareil. Sachant qu'un contrat a déjà été établi, le PD peut également utiliser le réseau (soit OTA ou OOB) pour configurer le DBP.

13.7 Diagrammes de transition d'état

Les options débattues jusqu'ici pour la configuration sont montrées ci-dessous dans des diagrammes de transition d'états.

La Figure 136 décrit les transitions d'état pertinentes pour la configuration pendant le cycle de vie d'un appareil de terrain. Le diagramme décrit des états en un site de fabrication et en un site utilisateur.



Légende

Anglais	Français
Assembled	Assemblé
OOB or OTA mechanism to install credentials, certificates, EUI-64	Mécanisme OOB ou OTA pour installer les authentifiants, les certificats, EUI-64
Edit config bits	Modifier les bits de configuration
Just built & flashed -has no key pair, no certificates -has default key K_global, join key & config bits preset to default	Juste construit et flashé - n'a aucune paire de clés, aucun certificat - a la clé de rattachement par défaut K_global et les bits de configuration préréglés aux valeurs par défaut

Anglais	Français
Un-provisioned -has been given its credentials; may have asymetric key pair and certificates -has default key K_global, join key & config bits preset to default	Non configuré - a reçu des authentifiants, peut avoir une paire de clés asymétriques et des certificats - a la clé de rattachement par défaut K_global et les bits de configuration préréglés aux valeurs par défaut
Factory pre-provisioned -has key pair and certificates -has either modified join key K_join and/or config bits changed	Préconfiguré en usine - a une paire de clés et des certificats -a soit la clé de rattachement K_join modifiée et/ou les bits de configuration changés
Shipping a fully compliant default device	Expédition d'un appareil par défaut totalement conforme
Insert secret join key S, OTA or OOB	Insérer clé de rattachement secrète S, OTA ou OOB
Shipping a device for a specific user group	Expédition d'un appareil pour un groupe spécifique d'utilisateurs
Manufacturer actions/site	Actions/site du fabricant
User actions/site	Actions/site de l'utilisateur
FACTORY DEFAULT	VALEURS D'USINE PAR DEFAUT
A1 Use OOB to edit or add keys, config, and other target network specifics	A1 Utiliser OOB pour modifier ou ajouter des clés, la configuration, et autres spécificités de réseau-cible
A3 Join-open (have K_join, but use K_global)	A3 Rattachement ouvert (a K_join, mais utilise K_global)
A3 Join-open (do not have K_join)	A3 Rattachement ouvert (n'a pas de K_join)
Idle, offline, MAY possess an altered K_Join	Inactif, hors ligne, PEUT posséder une K_Join altérée
Leave or power off or timeout	Quitter ou éteindre ou temporiser
Joined unsecured provisioning network; DO NOT have secret K_Join	A rejoint un réseau de configuration non sécurisé, N'A PAS de K_Join secrète
A2 Receive from SM secret K_Join, OTA, encrypted by my public key	A2 Recevoir du SM la K_Join secrète, OTA, chiffrée par ma clé publique
Join-target sym-key (have K_Join)	Clé symétrique de rattachement cible (posséder K_Join)
Request to sign a challenge with private key	Demande de signer un défi avec clé privée
Read signature, send certificates OTA	Lire signature, envoyer certificats OTA
Keys revoked	Clés révoquées
Join-target asym-key (have certificate)	Clé asymétrique de rattachement cible (posséder certificat)
TRUSTED and joined unsecured provisioning network; DO possess secret K_Join	FIABLE et a rejoint un réseau de configuration non sécurisé; possède EFFECTIVEMENT K_Join secrète
Reset to factory default	Réinitialisation aux valeurs d'usine par défaut
TRUSTED and joined the (secured) target network	APPROUVE et a rejoint le réseau cible (sécurisé)
Join target	Rejoindre cible
Receive non-secret target network specifics	Recevoir spécificités non secrètes de réseau-cible

Anglais	Français
Share certificates, share signatures, request to sign a challenge, all OTA in clear	Partager certificats, partager signatures, demander de signer un défi, tout cela par OTA en clair
Application messaging to change K_Join, change non-factory certificates, sign with private key, re-authenticate, edit config bits	Messagerie d'application pour changer K_Join, changer certificats autres que d'usine, signer avec clé privée, authentifier de nouveau, modifier les bits de configuration
State change can be disallowed through config bit A*n* in device, for *n* = 1..4	Le changement d'état peut être interdit par config bit A*n* dans l'appareil, pour *n* = 1..4

Figure 136 – Diagramme de transition d'états montrant les grandes lignes des étapes de configuration au cours du cycle de vie d'un appareil

Un appareil de terrain qui est nouvellement fabriqué passe à l'état "un-provisoned" lorsque son identité (par exemple, son adresse EUI64) et ses authentifiants lui sont fournis. Dans cet état, l'appareil a les valeurs de réglage par défaut telles que définies dans le Tableau 367.

Tableau 367 – Valeurs de réglage en usine par défaut

Attribut	Description	Valeur par défaut
Default symmetric join key K_global	La clé de rattachement symétrique utilisée pour rejoindre un réseau par défaut	Spécifié en 7.2.2.2.
Open symmetric join key (K_join = K_open)	La clé de rattachement symétrique de 128 bits ouverte utilisée pour rejoindre un réseau de configuration, puis pour recevoir de nouvelles clés de rattachements à une cible	Spécifié en 7.2.2.2
Allow OOB provisioning (A1)	Ce bit de configuration permet l'utilisation de mécanismes OOB pour configurer l'appareil. Ce bit n'est pas pertinent si l'appareil n'a aucun moyen OOB pour la configuration	1 = autorisé
Allow asymmetric-key-based provisioning (A2)	Ce bit de configuration permet l'utilisation de chiffrement asymétrique pour la configuration par OTA de K_join. Ce bit n'est pas pertinent si l'appareil ne prend pas en charge la cryptographie asymétrique	1 = autorisé
Allow default join (A3)	Ce bit de configuration permet à l'appareil de rejoindre un réseau par défaut. Certains appareils peuvent choisir de ne pas permettre du tout un rattachement par défaut	1 = autorisé
Allow reset to factory defaults (A4)	Ce bit de configuration permet l'exécution de commandes OTA qui réinitialise l'appareil à la configuration en usine par défaut	1 = autorisé

Le fabricant peut expédier des appareils avec ces valeurs de réglage par défaut.

En variante, l'appareil peut être préconfiguré pour un utilisateur particulier au site de fabrication. Lorsqu'un appareil est préconfiguré, les réglages par défaut de l'appareil sont changés. L'appareil peut avoir reçu une clé de rattachement symétrique à une cible qui est spécifique à un réseau cible au site utilisateur. En outre, n'importe lesquels des bits de configuration (A1, A2, A3 et A4) peuvent être changés. Par exemple, le rattachement au réseau par défaut et la réinitialisation aux valeurs d'usine par défaut peuvent être désactivés (A3, A4 = 0). Un tel appareil ne doit pas être capable d'être configuré par l'intermédiaire de la clé de rattachement symétrique ouverte (K_join= K_open).

L'appareil arrive au site utilisateur soit préconfiguré, soit avec les valeurs d'usine par défaut et il est dans l'état "idle".

Au site utilisateur, un appareil peut être configuré à l'aide des mécanismes d'OOB (A1 activé) avec une clé de rattachement symétrique et les informations relatives à un réseau pour rejoindre le réseau-cible. En variante, l'appareil peut avoir des clés de rattachement préinstallées et/ou les informations relatives à un réseau établies par le fabricant d'appareil. Si les informations relatives à un réseau ne sont pas configurées dans l'appareil à la fabrication, l'appareil peut rejoindre un réseau de configuration avec la clé de rattachement symétrique par défaut K_global spécifiée en 7.2.2.2 (si A3 est activé) et peut être configuré avec les informations relatives à un réseau en utilisant des mécanismes par liaison radio.

Les appareils qui échouent à rejoindre le réseau cible en utilisant leurs informations configurées peuvent chercher à rejoindre un réseau de configuration (si A3 est activé) en utilisant K_global. Après s'être joint en utilisant la clé de rattachement par défaut (K_global), le PD peut utiliser la méthode Write_Symmetric_Join_Key pour mettre à jour la K_join seulement si elle est envoyée chiffrée avec la clé de asymétrique du DBP.

Si l'appareil dans un état "idle" n'a pas une clé de rattachement symétrique installée valide et est autorisé à rejoindre un réseau par défaut, et A4 est activé, l'appareil doit lancer une recherche par balayage d'annonces afin d'atteindre un gestionnaire de sécurité/gestionnaire de système d'un réseau par défaut dans son voisinage.

Si une annonce est trouvée et l'appareil a des capacités cryptographiques asymétriques et des certificats PKI, il doit transmettre ses authentifiants au gestionnaire de sécurité associé au routeur d'annonce. Les routeurs d'annonce doivent transmettre des demandes de rattachement vers leurs gestionnaires de sécurité en utilisant un contrat établi que le routeur de publicité a conclu avec le gestionnaire de sécurité/gestionnaire de système.

Lorsque le gestionnaire de sécurité reçoit de nouveaux authentifiants d'appareil, il vérifie d'abord si, oui ou non, les appareils avec ces authentifiants sont attendus et autorisés pour le réseau cible. Cela peut être accompli par l'intermédiaire de la consultation dans les listes blanches pré-peuplées avec l'adresse EUI64I de l'appareil individuel. Les authentifiants d'appareil sont utilisés par le gestionnaire de système pour décider de la CA (et sa clé de rattachement asymétrique) à utiliser dans des étapes d'authentification ultérieures.

Si l'appareil est autorisé, l'authenticité des authentifiants est vérifiée par le gestionnaire de système. Les authentifiants d'appareil comprennent les certificats de l'appareil. En utilisant plusieurs certificats, le contrôle sur des données d'appareil peut[13] être constitué de deux cryptographiques asymétriques, l'une en utilisant la clé publique de la CA qui est déjà présente à l'intérieur du gestionnaire de sécurité (le PD) pour lire le premier certificat (appelé le certificat d'émetteur) et donc la clé publique de l'émetteur, suivie du deuxième certificat (appelé le certificat d'appareil) et donc la clé publique de l'appareil, en utilisant la clé de publique de l'émetteur. Une fois que la clé publique de l'appareil est obtenue, un mécanisme de défi/réponse (voir 7.4.6) est utilisé par le PD pour établir l'authenticité du DBP.

Une copie des données échangées dans les étapes précédentes peut être journalisée dans des dossiers publics dans le PD pour des besoins d'audit futur.

L'entrée d'utilisateur pour accepter l'appareil peut être sollicitée avant que l'appareil ne soit accepté. Une fenêtre de dialogue sur une interface homme-machine (IHM) connecté au gestionnaire de système peut chercher la confirmation qu'il convient d'autoriser l'appareil digne de confiance à rejoindre le réseau cible. Cela peut être un dialogue oui/non qui demande s'il convient qu'un appareil spécifique, avec une identité authentifiée spécifique, qui est un membre d'une famille d'appareils attendus et jugés bienvenus, soit en effet maintenant préparé pour rattachement sécurisé au réseau cible sécurisé. Lorsque cette étape d'entrée utilisateur est mise en œuvre, et la réponse d'utilisateur n'est pas reçue et aucune réponse n'est envoyée au cours de la période de temporisation de la réponse de rattachement, la demande de rattachement doit être considérée comme ayant échoué.

Si l'appareil est autorisé (présent dans la liste blanche) et authentique, le PD génère une nouvelle clé pour le DBP, la chiffre avec la clé asymétrique de DBP et l'émet vers le DBP. Une copie peut être journalisée dans des dossiers dans le PD pour des besoins d'audit futur.

L'échec dans l'une des étapes ci-dessus peut être dû à la perte de connectivité, aux temporisations, ou au déni de demande de rattachement provenant du DBP. Les exemples du dernier cas comprennent un statut négatif sur des listes blanches, une discordance pendant l'authentification, ou un rejet issu d'une fenêtre de dialogue sur une IHM. Lorsqu'il est clair

---

13 La chaîne à deux certificats décrite ici est seulement l'une des nombreuses topologies de certificats possibles avec plusieurs certificats. Le DPO fournit des attributs pour inclure plusieurs certificats.

qu'il convient de rejeter un DBP pour n'importe lesquelles de ces raisons, une alerte est générée par le gestionnaire de sécurité. Aucune réponse de rattachement ne doit être renvoyée à l'appareil indiquant un échec de rattachement à l'appareil.

Si le DBP n'a pas les modules cryptographiques asymétriques, mais il a la clé de rattachement symétrique ouverte, il peut rejoindre un réseau de configuration avec la clé de rattachement symétrique ouverte (K_join = K_open). Le droit d'accepter ou de rejeter la configuration des DBP qui utilisent les clés de rattachement symétrique ouvertes (K_join = K_open) est du ressort du PD. Par défaut, le PD ne doit pas configurer les appareils qui s'attachent avec la clé de rattachement ouverte; le PD peut toutefois être configuré pour configurer ces appareils. Si le PD est configuré pour permettre la configuration OTA ouverte, le DBP sera configuré avec une nouvelle clé de rattachement K_join pour rejoindre le réseau cible. Une fois qu'il est configuré, l'appareil ne doit pas réutiliser la clé ouverte à moins qu'il n'ait été réinitialisé aux valeurs d'usine par défaut (A4 est activé).

Une fois configuré, l'appareil peut continuer à rejoindre le réseau cible avec ses informations configurées. Comme partie intégrante du processus de rattachement de sous-réseau, l'appareil reçoit une clé principale, des clés T et des clés D, en plus d'établir un contrat avec le gestionnaire de système/sécurité du réseau-cible; le fonctionnement normal du réseau sécurisé normalisé suit.

Comme partie intégrante du fonctionnement normal d'un réseau, le gestionnaire de système du réseau peut configurer l'appareil avec des informations suffisantes pour rejoindre un autre réseau lorsque l'appareil quitte le réseau courant. Ce processus permet à un appareil de rejoindre et quitter plusieurs réseaux. La configuration pour un autre réseau en utilisant un réseau cible courant est accomplie comme suit.

a) Le DPSO dans le gestionnaire de système courant récupère des informations de réseau et des clés de sécurité de réseau issues du gestionnaire de système/gestionnaire de sécurité de l'autre réseau.

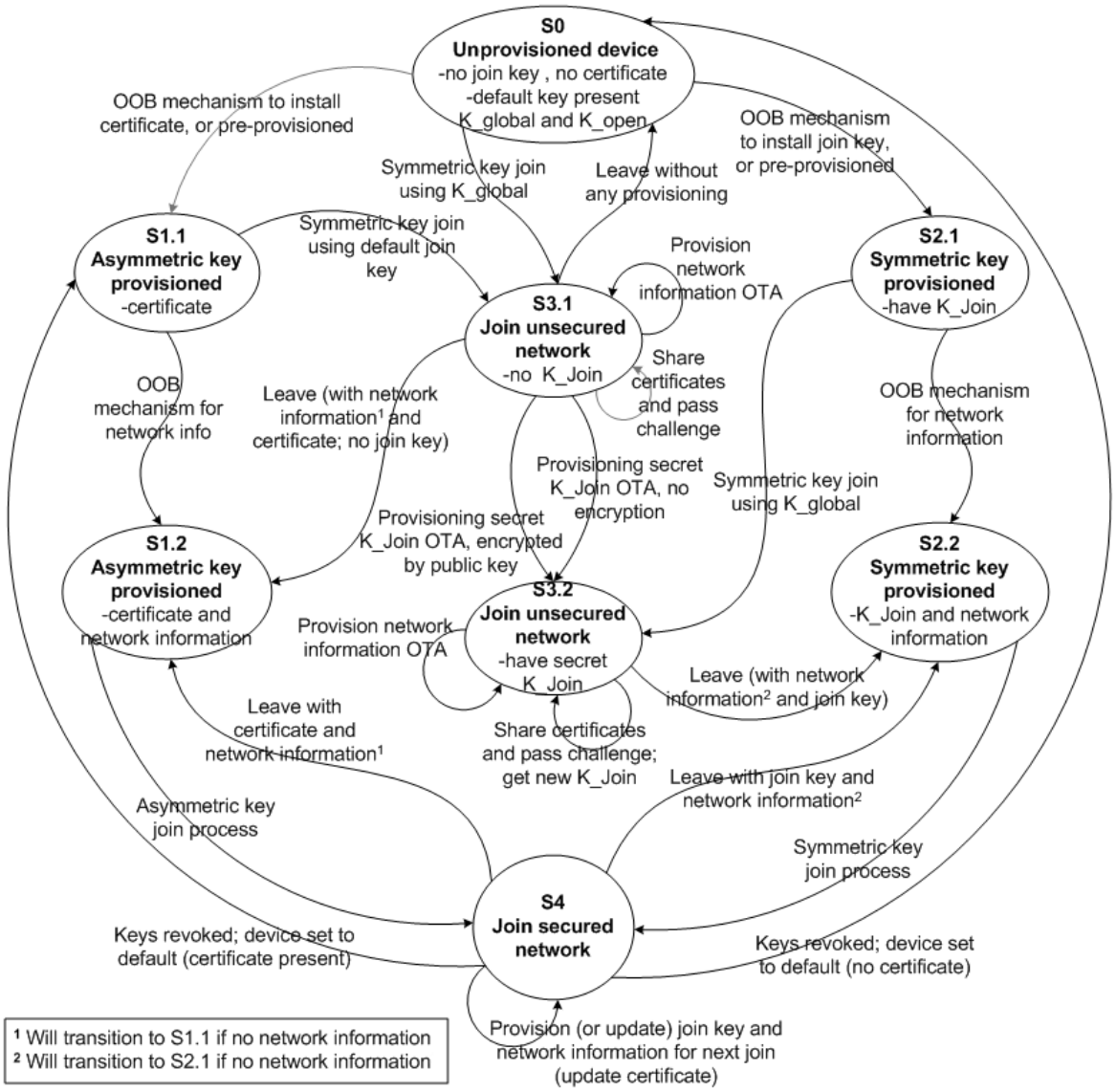
   NOTE L'interface pour une telle communication entre gestionnaires ne relève pas du domaine d'application de la présente norme.

b) Le DPSO dans le gestionnaire de système courant installe les informations dans le DPO de l'appareil.

c) Le DBP quitte le réseau courant.

d) Lorsque l'appareil quitte le réseau courant, il rejoint le prochain réseau avec l'information de réseau et de sécurité installée dans son DPO.

Comme décrit dans le présent document, il y a plusieurs chemins (et transitions d'états) disponibles pour un appareil non configuré pour être configuré et pour rejoindre finalement un réseau sécurisé. Ces chemins sont montrés par l'intermédiaire du diagramme de transitions d'états à la Figure 137. La Figure 137 est reliée (et équivalente) à la Figure 136; toutefois, la Figure 137 est décrite du point de vue d'un état interne d'appareil.



Légende

Anglais	Français
S0 Un-provisioned device -no join key , no certificate -default key present K_global and K_open	S0 Appareil non configuré - aucune clé de rattachement, aucun certificat - clé par défaut présente K_global et K_open
OOB mechanism to install certificate, or pre-provisioned	Mécanisme OOB pour installer le certificat, ou préconfiguré
OOB mechanism to install join key, or pre-provisioned	Mécanisme OOB pour installer la clé de rattachement certificat, ou préconfiguré
Symmetric key join using K_global	Clé de rattachement symétrique utilisant K_global
Leave without any provisioning	Quitter sans configuration
S1.1 Asymmetric key provisioned -certificate	S1.1 Clé asymétrique configurée -certificat
Symmetric key join using default join key	Rattachement à clés symétriques utilisant une clé de rattachement par défaut
Provision network information OTA	Configurer informations réseau par OTA
S2.1 Symmetric key provisioned -have K_Join	S2.1 Clé symétrique configurée - a K_Join

Anglais	Français
S3.1 Join unsecured network -no K_Join	S3.1 Rejoindre un réseau non sécurisé - pas de K_Join
OOB mechanism for network info	Mécanisme OOB pour infos réseau
Leave (with network information[a] and certificate; no join key)	Quitter (avec informations réseau[a] et certificat; aucune clé de rattachement)
Share certificates and pass challenge	Partager les certificats et réussir le défi
OOB mechanism for network information	Mécanisme OOB pour informations réseau
Provisioning secret K_Join OTA, no encryption	Configuration de K_Join secrète par OTA, aucun chiffrement
Symmetric key join using K_global	Rattachement à clés symétriques utilisant K_global
Provisioning secret K_Join OTA, encrypted by public key	Configuration de K_Join secrète par OTA, chiffrée par clé publique
S1.2 Asymmetric key provisioned -certificate and network information	S1.2 Clé asymétrique configurée -certificat et informations réseau
S2.2 Symmetric key provisioned -K_Join and network information	S2.2 Clé symétrique configurée -K_Join et informations réseau
S3.2 Join unsecured network -have secret K_Join	S3.2 Rejoindre réseau non sécurisé - avoir K_Join secrète
Provision network information OTA	Configurer informations réseau par OTA
Leave (with network information[b] and join key)	Quitter (avec informations [b] réseau et clé de rattachement)
Leave with certificate and network information[a]	Quitter avec certificat et informations[a] réseau
Share certificates and pass challenge; get new K_Join	Partager les certificats et réussir le défi; obtenir nouvelle K_Join
Leave with join key and network information [b]	Quitter avec clé de rattachement et information [b] réseau
Asymmetric key join process	Processus de rattachement à clés asymétriques
Symmetric key join process	Processus de rattachement à clés symétriques
S4 Join secured network	S4 Rejoindre réseau sécurisé
Keys revoked; device set to default (certificate present)	Clés révoquées, appareil mis aux valeurs par défaut (certificat présent)
Keys revoked; device set to default (no certificate)	Clés révoquées, appareil mis aux valeurs par défaut (aucun certificat)
[1] Will transition to S1.1 if no network information[1] [2] Will transition to S2.1 if no network information[2]	[1] Passera à S1.1 si aucune information réseau[1] [2] Passera à S2.1 si aucune information réseau[2]
Provision (or update) join key and network information for next join (update certificate)	Configurer (ou mettre à jour) la clé de rattachement et les informations réseau pour le prochain rattachement (mise à jour de certificat)

Figure 137 – Diagramme de transitions d'états montrant divers chemins pour rejoindre un réseau sécurisé

Les transitions et les chemins adressés à la Figure 137 comprennent:

a) Configuration OOB de clé symétrique et d'informations de réseau:

   1) Transitions d'états: S0 → S2.1 → S2.2 → S4.

   2) Synopsis: Des mécanismes OOB sont utilisés pour configurer un appareil avec la clé de rattachement à un réseau-cible (S0 –> S2.1) et des informations de réseau (S2.1 → S2.2). Ensuite, l'appareil utilise la procédure de rattachement symétrique (S2.2 → S4) pour rejoindre le réseau sécurisé.

b) Préconfiguré en usine (OOB ou autrement): Clés asymétriques et certificats et configuration OOB d'informations de réseau:

   1) Transitions d'états: S0 → S1.1 → S1.2 → S4.

   2) Synopsis: Un appareil est préconfiguré en usine avec des clés asymétriques et des certificats (S0 → S1.1). L'appareil a les informations nécessaires pour initier une

procédure de rattachement à clés asymétriques. Cependant, il n'a pas assez d'informations relatives à un réseau. Ces informations sont configurées en utilisant le mécanisme OOB (S1.1 → S1.2). Ensuite, l'appareil utilise la procédure de rattachement asymétrique pour rejoindre le réseau sécurisé (S1.2 → S4).

c) Configuration OOB d'informations de clés symétriques et configuration par OTA des informations de réseau:

   1) Transitions d'états: S0 → S2.1 → S3.2 → S2.2 → S4.

   2) Synopsis: Un appareil est configuré en utilisant un mécanisme OOB (ou préconfiguré) avec la clé de rattachement symétrique pour le réseau-cible (S0 → S2.1). L'appareil rejoint alors un réseau de configuration par défaut en utilisant la clé de rattachement par défaut, K_global (S2.1 → S3.2). Le PD dans le réseau de configuration fournit les informations de réseau pour le réseau cible. L'appareil quitte le réseau de configuration (S3.2 → S2.2) et rejoint le réseau sécurisé (S2.2 → S4) en utilisant la procédure de rattachement symétrique.

d) Clés asymétriques et certificats préconfigurés en usine (OOB ou autrement) et configuration par OTA des clés symétriques:

   1) Transitions d'états: S0 → S1.1 → S3.1 → S3.2 → S2.2 → S4.

   2) Synopsis: Un appareil est préconfiguré en usine avec des clés asymétriques et des certificats (S0 → S1.1). L'appareil a les informations nécessaires pour initier une procédure de rattachement à clés asymétriques. Cependant, il ne peut pas rejoindre un réseau-cible qui ne prend pas en charge le processus de rattachement de sous-réseau asymétrique. L'appareil rejoint alors un réseau de configuration par défaut qui est différent du réseau cible en utilisant la clé de rattachement par défaut, K_global (S1.1 → S3.1). Comme partie intégrante de ce réseau de configuration, l'appareil échange ses authentifiants, passe avec succès par un mécanisme défi-réponse, et reçoit du PD la clé de rattachement au réseau cible, chiffrée avec la clé publique de l'appareil (S3.1 → S3.2). L'appareil est alors configuré avec les informations de réseau par OTA. Ensuite, l'appareil quitte le réseau de configuration (S3.2 → S2.2) et rejoint le réseau sécurisé (S2.2 → S4) en utilisant la procédure de rattachement symétrique.

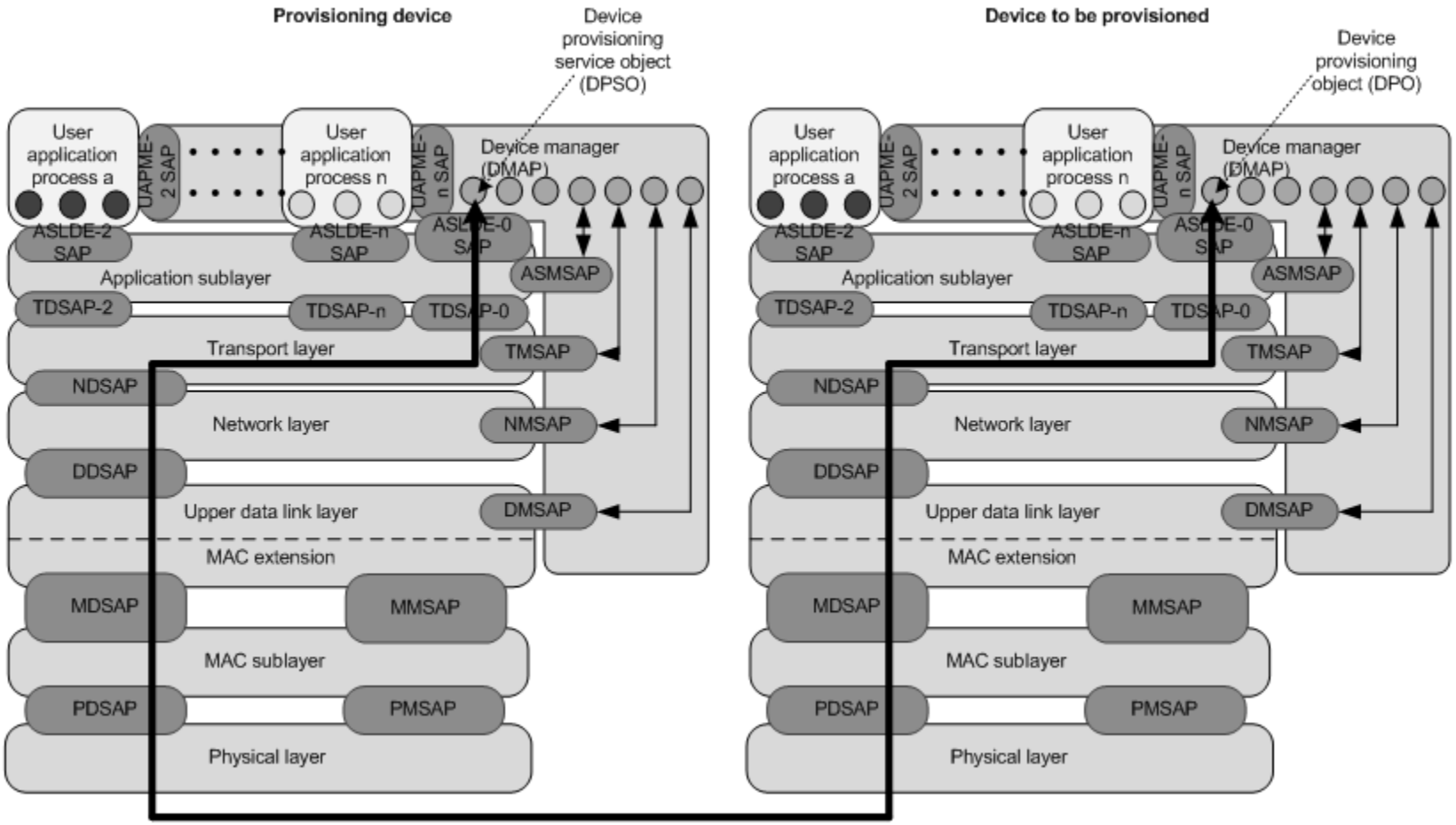
e) Configuration basée sur clé de rattachement ouverte en clair:

   1) Transitions d'états: S0 → S2.1 → S2.2 → S4(1) → S2.2 → S4(2).

   2) Synopsis: Un appareil qui a la clé de rattachement symétrique ouverte par défaut. Il utilise la procédure à clé de rattachement symétrique pour rejoindre un réseau de configuration (S2.2. → S4(1)). Comme partie intégrante de ce réseau de configuration, l'appareil est configuré avec la clé de rattachement au réseau cible et les informations de réseau. L'appareil quitte ensuite le réseau de configuration (S4(1) → S2.2). L'appareil est maintenant configuré pour rejoindre le réseau-cible; il rejoint le réseau-cible sécurisé en utilisant le processus de rattachement de sous-réseau à clé symétrique (S2.2 → S4(2)). Dans cette transition, la première fois que l'appareil a rejoint un réseau de configuration est indiquée par l'état S4(1), et la deuxième fois qu'il est rattaché au réseau cible est indiquée par l'état S4(2). Après que l'appareil a atteint S4(2), l'appareil ne peut pas utiliser la clé de rattachement symétrique ouverte, à moins qu'il soit réinitialisé aux valeurs d'usine par défaut.

13.8 Objets de protocole d'application de gestion d'appareil utilisés au cours de la configuration

La présente norme utilise un objet de DMAP et un objet SMAP au cours de la configuration. L'objet de configuration d'appareil (DPO) détient les réglages de configuration. La Figure 138 montre des objets de configuration et les interactions entre eux.



Légende

Anglais	Français
Provisioning device	Appareil de configuration
Device provisioning service object (DPSO)	Objet service de configuration d'appareil (DPSO)
Device to be provisioned	Appareil à configurer
Device provisioning object (DPO)	Objet de configuration de l'appareil (DPO)
User application process a	Processus d'application utilisateur a

Anglais	Français
User application process n	Processus d'application utilisateur n
Device manager (DMAP)	Gestionnaire d'appareil (DMAP)
Application sub-layer	Sous-couche d'application
Transport layer	Couche transport
Network layer	Couche réseau
Upper data link layer	Couche liaison de données supérieure
MAC extension	Extension MAC
MAC sub-layer	Sous-couche MAC
Physical layer	Couche physique

Figure 138 – Objets de configuration et interactions

Qu'il soit le gestionnaire de système/gestionnaire de sécurité dans un appareil tenu à la main ou le gestionnaire de système du réseau cible, le PD doit mettre en œuvre un objet de service de configuration d'appareil (DPSO) avec des attributs et des méthodes pour configurer le DBP. Le DPSO peut avoir une liste de clés symétriques, utilisée pour configurer des appareils qui n'ont pas de clés préinstallées.

La liste blanche, les informations de clés symétriques, et les informations de réseau-cible dans le DPSO peuvent être maintenues avec des informations spécifiques à un appareil dans l'attribut White_List_Array dans le Tableau 372. En variante, un groupe de clés symétriques valides peut être maintenue.

Lorsque le DBP rejoint le réseau de configuration en utilisant K_global, un contrat est établi entre le PD et le DBP. Le DPSO dans le PD peut utiliser le contrat établi pour communiquer avec le DPO dans le DBP. Les primitives des services read et write sont utilisées pour accéder aux attributs du DPO et les positionner. Un sous-ensemble d'informations de réseau et de confiance peut maintenant être configuré dans le DPO en utilisant le DPSO. Pour écrire la nouvelle clé de rattachement symétrique dans l'appareil, le DPSO invoque la méthode de Write_Join_key du DPO. Cette méthode est permise si la nouvelle valeur de clé a été reçue sous la protection de la cryptographie asymétrique. Les attributs dans le DPO comprennent des informations tant relatives à un réseau que relatives à la confiance.

Il convient que les utilisateurs qui désirent plus de sécurité pendant la configuration utilisent l'authentification par cryptographie asymétrique et la technique de chargement de clés sécurisées pendant les étapes relatives à la confiance. En variante, des mécanismes hors bande peuvent être utilisés pour configurer des clés de rattachement.

Une fois que les informations appropriées relatives à la confiance et à un réseau ont été configurées dans le DPO, l'appareil est prêt à rejoindre le réseau cible. Le réseau de configuration peut également être utilisé pour la configuration d'appareil. Sachant qu'un contrat a été déjà établi, le PD peut également utiliser le réseau (ou des moyens OOB) pour configurer les objets UAP et DMAP appropriés du DBP.

L'objet de configuration d'appareil (DPO) fournit un attribut appelé le Target_DL_Config dans le format de DL_Config_Info. DL_Config_Info est décrit à l'Article 9 (voir Tableau 102) pour configurer divers attributs de la DL. Une fois qu'il est configuré avec cet attribut, le DPO fournit à la DL un OctetString encapsulant DL_Config_Info qui inclut au moins une supertrame, et au moins une liaison, qui peuvent être utilisées par la DL pour rechercher des annonces. Des modèles d'intervalles de temps spécifiques à un réseau cible (par exemple, autres que par défaut), des profils de sauts de voies, des supertrames et des liaisons peuvent également être fournis au DBP par l'attribut Target_DL_Config. Une telle configuration aide à réduire la quantité d'informations (supertrames de rattachement, par exemple) qui autrement seraient tenues d'être annoncées par les routeurs d'annonce de réseau cible.

La DL de l'appareil joue un rôle majeur au cours de la configuration et du rattachement de l'appareil. Le diagramme d'états de la DL lorsqu'elle passe par le processus de configuration est décrit en 9.1.14.2.

Si le processus de configuration est réussi, le DPO fournit à la DL le jeu d'attributs, y compris les informations relatives au sous-réseau D, les supertrames, et les liaisons, que la DL peut utiliser pour rechercher le réseau cible et le(s) sous-réseau(x) D correspondants. Dans l'état "provisoned", la DL actionne son diagramme d'états tel qu'il est configuré dans les supertrames et les liaisons qui ont été fournies par le DPO. Le fonctionnement de la supertrame peut être retardé ou désactivé en positionnant le champ IdleTimer dans la supertrame.

Sachant que l'appareil retient les informations utilisées pour configurer la DL (tous les attributs du DPO), cela assure un moyen de réinitialiser la DL à son état "provisioned" en mettant la DL dans son état par défaut et en ajoutant ensuite les attributs configurés.

Le DPO doit être accessible au gestionnaire de système du réseau cible après s'être rattaché avec Key_Join. Une fois que l'appareil rejoint un réseau cible, le gestionnaire de système du réseau cible a la capacité de changer les attributs du DPO. Le gestionnaire de système du réseau cible a la capacité d'ordonner à l'appareil de rejoindre un autre réseau cible en fournissant des informations de réseau et de confiance de l'autre réseau. Selon la valeur du bit de configuration A4, le gestionnaire de système du réseau cible a la capacité d'invoquer une méthode DPO.Reset_To_Defaults pour retirer de l'appareil les informations de confiance.

NOTE Dans la phase de configuration, le DPO dans le DBP est accessible par la fonctionnalité de gestionnaire de système/gestionnaire de sécurité dans le PD.

13.9 Objets de gestion

13.9.1 Objet de configuration d'appareil

Le Tableau 368 décrit les attributs du DPO. Le type de données, la valeur par défaut, et une courte description sont fournis pour chaque attribut. Chaque attribut a également l'accessibilité de lecture seule ou de lecture/écriture. Les attributs du DPO sont accessibles seulement au gestionnaire de système/gestionnaire de sécurité. La valeur d'un attribut en lecture seule peut être positionnée seulement au moment de la fabrication d'appareil (à savoir à un moment avant que l'appareil soit certifié) ou en interne par l'appareil; aucune entité externe à l'appareil ne peut changer cet attribut. L'accessibilité en lecture/écriture implique que les entités externes à l'appareil peuvent changer la valeur de l'attribut. Les attributs du DPSO sont seulement accessibles (lecture/écriture) au gestionnaire de système.

La valeur des attributs constants n'est pas changée pendant le cycle de vie de l'appareil, ni intérieurement ni extérieurement. La définition de la classification est donnée en 12.6.3.

Tableau 368 – Objet de configuration d'appareil *(1 de 7)*

Nom du type d'objet normalisé: Device provisioning object (DPO, objet de configuration d'appareil)				
Identificateur du type d'objet normalisé: 120				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Default_NWK_ID	1	Une identification réseau publiée pour le réseau par défaut	Type: Unsigned16 Classification: Constant Accessibilité: Lecture seule (Read Only) Valeur par défaut: 0x0001	Il s'agit de l'identification réseau pour le réseau par défaut. Le réseau par défaut peut être utilisé pour former le mini-réseau de configuration
Default_SYM_Join_Key	2	Une clé de rattachement publiée du réseau par défaut	Type: Clé symétrique (SymmetricKey) Classification: Constant Accessibilité: Lecture seule (Read Only) Valeur par défaut: K_global, 7.2.2.2	Cette clé est utilisée par des appareils pour rejoindre le réseau par défaut. Les clés par défaut peuvent être utilisées pour former le mini-réseau de configuration
Open_SYM_Join_Key	3	Une clé de rattachement publiée du réseau par défaut	Type: Clé symétrique (SymmetricKey) Classification: Constant Accessibilité: Lecture seule (Read Only) Valeur par défaut: K_open, 7.2.2.2	Cette clé est utilisée par des appareils pour rejoindre le réseau de configuration non sécurisé

Tableau 368 *(2 de 7)*

Nom du type d'objet normalisé: Device provisioning object (DPO, objet de configuration d'appareil)				
Identificateur du type d'objet normalisé: 120				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Default_Channel_List	4	La liste des voies 2,4 GHz utilisées par le réseau par défaut. L'attribut est codé comme un matriciel de 16 bits pour représenter les 16 fréquences	Type: Unsigned16 Classification: Constant Accessibilité: Lecture seule (Read Only) Valeur par défaut: 0x7FFF	La liste des voies utilisées par les routeurs d'annonce du réseau par défaut. Pour rejoindre le réseau par défaut, l'appareil peut recevoir des annonces sur n'importe laquelle des fréquences
Join_Method_Capability	5	Les capacités d'attachement de l'appareil	Type: Unsigned2 Classification: Constant Accessibilité: Lecture/écriture Valeur par défaut: 00 Valeurs nommées: 00= rattachement par défaut seulement 01= rattachement à clés symétriques seulement 10= rattachement à clés asymétriques seulement 11= tout rattachement à clés	Cet attribut définit les capacités d'attachement d'un appareil. L'appareil peut utiliser soit les clés symétriques, soit l'infrastructure à clés asymétriques pour rejoindre un réseau cible. Cet attribut définit simplement les capacités de l'appareil. La véritable méthode utilisée pour rejoindre le réseau cible doit être définie par le PD
Allow_Provisioning	6	Une valeur booléenne définie pour indiquer si un appareil peut ou non être configuré	Type: Boolean1 Classification: Statique Accessibilité: Lecture/écriture Valeur par défaut: vrai	Cet indicateur est utilisé pour verrouiller l'état d'un appareil déjà configuré. Si cette valeur est mise, l'appareil n'acceptera aucune lecture ou écriture dans les attributs de réseau cible

Tableau 368 *(3 de 7)*

Nom du type d'objet normalisé: Device provisioning object (DPO, objet de configuration d'appareil)				
Identificateur du type d'objet normalisé: 120				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Allow_Over_The_Air_ Provisioning	7	Une valeur booléenne définie pour indiquer si un appareil peut ou non être configuré	Type: Boolean1 Classification: Statique Accessibilité: Lecture/écriture Valeur par défaut: vrai	Ce booléen indique si la configuration par liaison radio est activée ou désactivée. Si la configuration par liaison radio est désactivée, l'appareil doit être configuré avec une méthode hors bande. Cette valeur doit être définie sur false pour les appareils dorsaux. Dans tous les cas, la configuration n'est autorisée que si l'attribut Allow_Provisioning est activé
Allow_OOB_Provisioning	8	Une valeur booléenne définie pour indiquer si un appareil peut ou non être configuré à l'aide de moyens OOB	Type: Boolean1 Classification: Statique Accessibilité: Lecture/écriture Valeur par défaut: vrai	Le booléen permet d'empêcher les appareils d'accepter des informations de configuration par des moyens OOB
Allow_Reset_to_Factory_ Defaults	9	Une valeur booléenne définie pour indiquer si un appareil peut ou non être réinitialisé à ses valeurs d'usine par défaut	Type: Boolean1 Classification: Statique Accessibilité: Lecture/écriture Valeur par défaut: vrai	Ce booléen permet de bloquer la réinitialisation des appareils à leurs valeurs par défaut d'usine par un gestionnaire de système
Allow_Default_Join	10	Une valeur booléenne définie pour indiquer si un appareil peut s'attacher à un réseau à l'aide des clés par défaut	Type: Boolean1 Classification: Statique Accessibilité: Lecture/écriture Valeur par défaut: vrai	Ce booléen est utilisé pour forcer les appareils à s'attacher à un réseau cible en particulier et à ne pas s'attacher à n'importe quel réseau par défaut. Les appareils qui choisissent de ne pas s'attacher à un réseau par défaut peuvent définir cet attribut sur false

Tableau 368 *(4 de 7)*

Nom du type d'objet normalisé: Device provisioning object (DPO, objet de configuration d'appareil)				
Identificateur du type d'objet normalisé: 120				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Target_NWK_ID	11	L'ID réseau du réseau cible auquel cet appareil est configuré pour s'attacher	Type: Unsigned16 Classification: Statique Accessibilité: Lecture/écriture Valeur par défaut: 0	Cet attribut indique le réseau cible que cet appareil doit rejoindre [a]
Target_NWK_BitMask	12	Bitmask pour la correspondance des bits de l'ID du réseau cible	Type: Unsigned16 Classification: Statique Accessibilité: Lecture/écriture Valeur par défaut: 0xFFFF	Le bitmask est utile pour la correspondance de plusieurs réseaux cibles. Si la valeur d'un bit dans le bitmask est 1, le bit doit avoir une correspondance exacte avec le bit correspondant dans le réseau cible. La valeur par défaut des 1 partout indique que tous les bits de l'ID de réseau ont besoin de concorder
Target_Join_Method	13	Indique s'il convient que l'appareil utilise le mécanisme d'attachement de clé symétrique ou de clé asymétrique pour s'attacher au réseau cible	Type: Unsigned1 Classification: Statique Accessibilité: Lecture/écriture Valeur par défaut: 1 Valeurs nommées: 0: clé symétrique 1: clé asymétrique	Le paragraphe 7.4.4 définit deux méthodes différentes pour le rattachement en fonction de l'utilisation soit de clés symétriques, soit de certificats à clés asymétriques. Cet attribut établit la méthode devant être utilisée pour rejoindre un réseau cible
Target_Security_Manager_EUI	14	L'adresse EUI64 du gestionnaire de sécurité dans le réseau cible que l'appareil est censé rejoindre	Type: AdresseEUI64 Classification: Statique Accessibilité: Lecture/écriture Valeur par défaut: 0xFF…FF (all 0xFF)	Définit sur EUI64Address pour le gestionnaire de sécurité que l'appareil est configuré à rejoindre

Tableau 368 *(5 de 7)*

Nom du type d'objet normalisé: Device provisioning object (DPO, objet de configuration d'appareil)				
Identificateur du type d'objet normalisé: 120				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Target_System_Manager_Address	15	L'adresse IPv6 du gestionnaire de sécurité/système dans le réseau cible que l'appareil est censé rejoindre	Type: IPv6Address Classification: Statique Accessibilité: Lecture/écriture	L'adresse IPv6 est exigée pour que des appareils dorsaux rejoignent le réseau. Les appareils dorsaux n'ont pas un routeur d'annonce – donc, une demande de rattachement est envoyée à l'adresse IPv6 du gestionnaire de système/gestionnaire de sécurité pour commencer le processus de rattachement. Les appareils E/S et les appareils de routage peuvent ne pas être configurés avec cet attribut
Target_Channel_List	16	La liste des voies utilisées par le réseau cible. L'attribut est codé comme un matriciel de 16 bits pour représenter les 16 fréquences	Type: BitArray16 Classification: Statique Accessibilité: Lecture/écriture	Le réseau cible peut être en train d'utiliser seulement un sous-ensemble de voies pour des annonces par les routeurs de rattachement. En utilisant seulement un sous-ensemble des fréquences, les appareils alimentés par batteries peuvent rapidement rejoindre le réseau cible en se mettant à l'écoute dans ce sous-ensemble de fréquences seulement
Target_DL_Config	17	Les informations de configuration de DL pour cet appareil	Type: OctetString Classification: Statique Accessibilité: Lecture / Ecriture	Cet attribut indique les différents paramètres de configuration pour le DL de l'appareil. La structure de cet attribut est définie dans DL_Config_Info définie à l'Article 9

Tableau 368 *(6 de 7)*

Nom du type d'objet normalisé: Device provisioning object (DPO, objet de configuration d'appareil)				
Identificateur du type d'objet normalisé: 120				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
PKI_Certificate_Type	18	(Option de cryptographie à clés asymétriques) Ce champ indique un type de certificat dans PKI_Root_Certificate et dans PKI_Certificates	Type: Unsigned8 Classification: Statique Accessibilité: Lecture/écriture Valeur par défaut: 0 Valeurs nommées: 0: certificat implicite; 1: certificat manuel	Ce champ indique un type de certificat dans PKI_Root_Certificate et PKI_Certificates
PKI_Root_Certificate	19	(Option de cryptographie à clés asymétriques) Le certificat racine de l'autorité de certification qui émet le certificat à l'appareil	Type: OctetString Classification: Statique Accessibilité: Lecture/écriture	Le certificat de racine de l'autorité de certification et sa clé asymétrique correspondante sont utilisés pour vérifier les certificats des nœuds d'homologues. Le certificat racine peut être mis à jour par le gestionnaire de système
Nombre de PKI_Certificates	20	(Option de cryptographie à clés asymétriques) Le nombre de certificats stockés dans l'attribut PKI_certificate	Type: Unsigned8 Classification: Statique Accessibilité: Lecture/écriture Valeur par défaut: 0	Ce champ indique le nombre de certificats disponibles dans l'attribut PKI_Certificate
PKI_Certificate	21	(Option de cryptographie à clés asymétriques) Le certificat délivré à cet appareil pour le rattachement en utilisant l'infrastructure à clés asymétriques	Type: Array of OctetString Classification: Statique Accessibilité: Lecture/écriture	Si Target_Join_Method est mis à Asymmetric-key, cet attribut contient le certificat (qui inclut la clé asymétrique, l'ID d'appareil, et autre texte) signé par une autorité de certification qui est exigé pour rejoindre le réseau cible

Tableau 368 *(7 de 7)*

Nom du type d'objet normalisé: Device provisioning object (DPO, objet de configuration d'appareil)				
Identificateur du type d'objet normalisé: 120				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Current_UTC_Adjustment	22	La valeur actuelle de l'ajustement de seconde intercalaire accumulée de l'UTC	Type: Integer16	Voir le Tableau 25, l'attribut 1 et la note de pied de page
			Classification: Statique	
			Accessibilité: Lecture/écriture	
			Valeur par défaut: 35	

[a] Si Target_NWK_BitMask (attribut 12) est défini sur 0xFFFF, l'appareil doit ignorer les annonces des routeurs qui appartiennent à un autre réseau que le réseau cible indiqué. Dans le cas contraire, une combinaison d'ID réseau et de bitmask doit être utilisée. (Voir la description de l'attribut 12 sur la façon dont le NetworkID et le bitmask sont combinés.) Cela aide avec les rattachements rapides et empêche également des appareils d'essayer de rejoindre tous les réseaux dans leur voisinage. Cette valeur peut être mise à 0 pour permettre des réponses à n'importe quel routeur d'annonce.

13.9.2 Méthodes et alertes d'objet de configuration d'appareil

Plusieurs méthodes et alertes sont disponibles dans le DPO. Le Tableau 369 décrit la méthode Reset_To_Default.

Tableau 369 – Méthode Reset_To_Default

Nom du type d'objet normalisé: Device provisioning object (DPO, objet de configuration d'appareil)				
Identificateur du type d'objet normalisé: 120				
Nom de méthode	**ID de méthode**	**Description de la méthode:**		
Reset_To_Default	1	Cette méthode est utilisée pour réinitialiser aux valeurs de réglage par défaut pour la configuration. Cette méthode ne doit être exécutée que quand Allow_Provisioning est activé		
	Input arguments (aucun)			
	Arguments de sortie			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	Etat	Unsigned8	Valeurs nommées: 0: réussite; autre: échec

Le Tableau 370 décrit la méthode pour écrire une clé de rattachement symétrique. La clé de rattachement ne doit être pas exposée à un appareil distant, et peut être exposée à un processus interne limité. La méthode DPO.Write_Symmetric_Join_Key() installe une clé de rattachement dans une zone de mémoire qui n'est pas utilisée pour des attributs (stockage sécurisé, par exemple).

Tableau 370 – Méthode d'écriture de clé de rattachement symétrique

<table>
<tr><th colspan="5">Nom du type d'objet normalisé: Device provisioning object (DPO, objet de configuration d'appareil)</th></tr>
<tr><th colspan="5">Identificateur du type d'objet normalisé: 120</th></tr>
<tr><th>Nom de méthode</th><th>ID de méthode</th><th colspan="3">Description de la méthode:</th></tr>
<tr><td rowspan="8">Write_SYM_join_key</td><td>2</td><td colspan="3">Cette méthode est utilisée pour écrire une clé d'attachement symétrique à un appareil. Cette méthode est évoquée par le DPSO pour configurer un DBP avec la clé de rattachement de Target. Selon la méthode de configuration utilisée, cette APDU d'appel de méthode et donc la clé de rattachement peuvent être chiffrées par la clé T entre l'appareil et le PD seul ou la clé asymétrique de l'appareil dans l'APDU en plus de l'APDU chiffrée par clé T. Cette méthode ne doit être exécutée que quand Allow_Provisioning est activé</td></tr>
<tr><td colspan="4">Arguments d'entrée</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td>1</td><td>New_Key_Value</td><td>Clé symétrique (SymmetricKey)</td><td>Nouvelle clé de rattachement devant être installée.</td></tr>
<tr><td>2</td><td>Encrypted By</td><td>Unsigned8</td><td>Valeurs nommées:
0: TL_Session_Key_Only,
1: Asymmetric_Key</td></tr>
<tr><td colspan="4">Arguments de sortie</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td>1</td><td>Etat</td><td>Unsigned8</td><td>Valeurs nommées:
0: réussite;
> 0: échec</td></tr>
</table>

13.10 Objet service de configuration d'appareil

13.10.1 Attributs de l'objet service de configuration d'appareil

Le Tableau 371 décrit les attributs du DPSO.

Le gestionnaire de système peut choisir soit les informations de configuration particulières pour chaque adresse EUI64, soit un jeu de clés de rattachement pour un ensemble d'adresses EUI64 sans mapping biunivoque. L'attribut Boolean1 DPSO.Enable_White_List_Array est utilisé pour spécifier quelle méthode est utilisée.

Dans le DPSO.Enable_White_List_Array mis, DPSO.White_List_Array est utilisé pour installer des informations de configuration particulières pour chaque adresse EUI64 du DBP.

Si DPSO.Enable_White_List_Array n'est pas mis, le PD doit vérifier s'il y a au moins autant d'entrées dans DPSO.SYM_Key_List que d'entrées dans DPSO.White_List. La présente norme ne spécifie pas comment chaque entrée dans DPSO.SYM_Key_List et DPSO.White_List est mise en mapping.

Tableau 371 – Objet service de configuration d'appareil *(1 de 4)*

Nom du type d'objet normalisé: Device provisioning service object (DPSO, objet service de configuration d'appareil ()				
Identificateur du type d'objet normalisé: 106				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
White_List	1	Une liste d'appareils autorisés à être configurés par cet objet	Type: Array of EUI64Address Classification: Statique Accessibilité: Lecture/écriture Valeur par défaut: [ ] -- empty	Cette liste contient les adresses EUI64 de l'appareil en cours de configuration. Cette liste peut être utilisée pour limiter un appareil de configuration à configurer seulement un ensemble d'appareils dont les adresses EUI64 sont dans cette liste. Si cette liste est vide, l'appareil de configuration peut configurer n'importe quel appareil
Symmetric_Key_List	2	Une liste de clés de rattachement valides avec lesquelles un appareil peut être configuré	Type: Array of SymmetricKey Classification: Statique Accessibilité: Lecture/écriture Valeur par défaut: {K_global] -- 7.2.2.2	Cette clé est utilisée par des appareils pour rejoindre le réseau cible avec une entropie suffisante
Symmetric_Key_Expiry_Times	3	Le temps d'expiration pour chaque clé	Type: Array of TAITimeRounded Classification: Statique Accessibilité: Lecture/écriture Valeur par défaut: [0xFFFF FFFF] [a]	Cet attribut établit le temps d'expiration de chacune des clés symétriques. La clé est seulement utilisée pour la configuration si elle n'a pas expiré
Target_NWK_ID	4	L'ID de réseau du réseau-cible que les appareils configurés par cet objet sont censés rejoindre	Type: Unsigned16 Classification: Statique Accessibilité: Lecture/écriture Valeur par défaut: 0	Cet attribut indique le réseau cible (ID de sous-réseau) qu'un appareil configuré doit rejoindre

Tableau 371 *(2 de 4)*

Nom du type d'objet normalisé: Device provisioning service object (DPSO, objet service de configuration d'appareil ()				
Identificateur du type d'objet normalisé: 106				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Target_Join_Method	5	Valeur booléenne pour indiquer s'il convient que les appareils configurés par cet objet utilisent le mécanisme d'attachement de clé symétrique ou de clé asymétrique pour s'attacher au réseau cible	Type: Unsigned8 Classification: Statique Accessibilité: Lecture/écriture Valeur par défaut: 0	L'Article 7 définit deux méthodes différentes pour le rattachement en fonction de l'utilisation de clés symétriques ou asymétriques. Cet attribut établit la méthode devant être utilisée pour rejoindre un réseau cible. Valeurs nommées: 0: clé symétrique; 1: clé asymétrique
Target_Security_Manager_EUI	6	L'adresse EUI64 du gestionnaire de sécurité dans le réseau cible que l'appareil configuré par cet objet est censé rejoindre	Type: AdresseEUI64 Classification: Statique Accessibilité: Lecture/écriture	Définit sur EUI64Address pour le gestionnaire de sécurité que l'appareil est configuré à rejoindre
Target_System_Manager_Address	7	L'adresse IPv6 du gestionnaire de sécurité/système dans le réseau cible que l'appareil configuré par cet objet est censé rejoindre	Type: IPv6Address Classification: Statique Accessibilité: Lecture/écriture Plage valide: toutes les valeurs avec réglage sur bit le plus élevé	L'adresse IPv6 est exigée pour que des appareils dorsaux rejoignent le réseau
Target_Channel_List	8	La liste des voies utilisées par le réseau par défaut. L'attribut est codé comme un matriciel de 16 bits pour représenter les 16 fréquences	Type: BitArray16 Classification: Statique Accessibilité: Lecture/écriture	Le réseau cible peut être en train d'utiliser seulement un sous-ensemble de voies pour des annonces par les routeurs de rattachement
Target_DL_Config	9	Les informations de configuration de DL pour que l'appareil soit configuré par cet objet	Type: OctetString Classification: Statique Accessibilité: Lecture / écriture	Cet attribut indique les différents paramètres de configuration pour le DL de l'appareil. La structure de cet attribut est définie dans DL_Config_Info définie à l'Article 9, dans le Tableau 102

Tableau 371 *(3 de 4)*

Nom du type d'objet normalisé: Device provisioning service object (DPSO, objet service de configuration d'appareil ()				
Identificateur du type d'objet normalisé: 106				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Allow_Provisioning	10	Une valeur booléenne définie pour indiquer si un appareil peut ou non être à nouveau configuré	Type: Boolean1 Classification: Statique Accessibilité: Lecture/écriture Valeur par défaut: vrai	Cet indicateur est utilisé pour verrouiller l'état futur d'un appareil configuré
Allow_Default_Join	11	Une valeur booléenne définie pour indiquer si un appareil configuré par cet objet peut s'attacher à un réseau à l'aide des clés par défaut	Type: Boolean1 Classification: Statique Accessibilité: Lecture/écriture Valeur par défaut: Not_allowed (0)	Cet indicateur est utilisé pour forcer les appareils configurés à s'attacher à un réseau cible en particulier et à ne pas s'attacher à un réseau par défaut. Une fois configuré, il convient que l'appareil s'attache au réseau cible
Enable_White_List_Array	12	Une valeur booléenne définie pour indiquer si l'objet de configuration est conçu pour définir des informations de configuration spécifiques pour l'appareil	Type: Boolean1 Classification: Statique Accessibilité: Lecture/écriture Valeur par défaut: faux	SI ce fanion est mis, le DPSO est capable de configurer des appareils différents (sur la base de l'adresse EUI 64) avec des informations de configuration différentes. Cela peut être utilisé pour configurer un appareil particulier, avec un gestionnaire de sécurité particulier et un Target_Join_Time particulier, par exemple
White_List_Array	13	Une matrice des adresses au format EUI que le DPSO vise à configurer avec les informations de configuration correspondantes pour l'appareil en question	Type: Array of DPSOWhiteListTbl Classification: Statique Accessibilité: Lecture/écriture	Cet attribut ne doit être utilisé que si Device_specific_provisioning_flag est défini. Il contient les informations de configuration spécifiques à l'appareil comme des clés de rattachement spécifiques à un appareil, un gestionnaire de sécurité cible spécifique à un appareil, etc.
White_List_Array_Meta	14	Métadonnées pour White List Array (Attribut 13) ou ensemble de White_List (Attribut 1) et SYM_Key_List (Attribut 2)	Type: Metadata_attribute Classification: Statique Accessibilité: Lecture seule (Read only)	Métadonnées contenant un compte du nombre d'entrées et la capacité (le nombre total de rangées autorisé) de la table White_List_Array ou de l'ensemble de White_List [b]

Tableau 371 *(4 de 4)*

Nom du type d'objet normalisé: Device provisioning service object (DPSO, objet service de configuration d'appareil ()				
Identificateur du type d'objet normalisé: 106				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
DPSO_Alerts_AlertDescriptor	15	Utilisé pour changer la priorité des alertes de DPSO qui appartiennent à la catégorie de sécurité; ces événements peuvent également être activés ou désactivés	Type: Alert report descriptor (descripteur de rapports d'alertes) Classification: Statique Accessibilité: Lecture/écriture Valeur par défaut: [faux, 6]	Voir la description des alertes au Tableau 374 et au Tableau 375
Current_UTC_Adjustment	16	La valeur actuelle de l'ajustement de seconde intercalaire accumulée de l'UTC	Type: Integer16 Classification: Statique Accessibilité: Lecture/écriture Valeur par défaut: 35	Voir le Tableau 25, l'attribut 1 et la note de pied de page

[a] Lorsqu'un délai d'expiration de clé calculé donne la valeur 0xFFFF FFFF comme résultat, il est augmenté (circulairement) à la valeur modulo 0x0000 0000, ainsi la valeur 0xFFFF FFFF peut être utilisée pour désigner une clé qui n'expire jamais.

[b] Si Enable_White_List_Array est activé, cet attribut spécifie un compte du nombre d'entrées et de la capacité pour White_List_Array. Si Enable_White_List_Array est désactivé, cet attribut spécifie un compte du nombre d'entrées et de la capacité pour l'ensemble de White_List et de SYM_Key_List.

13.10.2 Attributs structurés de l'objet service de configuration d'appareil

Le Tableau 372 décrit les attributs structurés du DPSO. White_List_Array est utilisé lorsque le PD a des informations spécifiques à un appareil, c'est-à-dire, si le PD a des clés de rattachement symétriques et d'autres attributs qui sont spécifiques à un appareil. Dans ce cas, il est exigé une matrice structurée qui stocke les informations de configuration indexées par l'identificateur d'adresse EUI64 du DBP. Cette matrice indexée est décrite dans le Tableau 372.

Après que le PD a reçu l'attribut Device_SYM_Key provenant du gestionnaire de sécurité, le PD ne doit pas dévoiler l'attribut Device_SYM_Key à l'extérieur.

NOTE L'interface entre le PD et le gestionnaire de sécurité ne relève pas du domaine d'application de la présente norme.

Tableau 372 –Structure de données DPSOWhiteListTbl

Nom du type de données normalisé: DPSOWhiteListTbl		
Code du type de données normalisé: 440		
Nom de l'élément	**Identificateur de l'élément**	**Type de l'élément scalaire**
Device_EUI	1	Type: Array of EUI64Address
		Classification: Statique
		Accessibilité: Lecture seule (Read only)
		Valeur par défaut: [ ] -- empty
Device_Tag	2	Type: Array of VisibleString
		Classification: Statique
		Accessibilité: Lecture/écriture
		Valeur par défaut: [""]
Symmetric_Key_List	3	Type: Array of SymmetricKey
		Classification: Statique
		Accessibilité: Lecture seule (Read only)
		Valeur par défaut: {K_global] -- 7.2.2.2
Symmetric_Key_Expiry_Times	4	Type: Array of TAIRounded
		Classification: Statique
		Accessibilité: Lecture seule (Read only)
		Valeur par défaut: [0xFFFF FFFF] [a]
Target_NWK_ID	5	Type: Unsigned16
		Classification: Statique
		Accessibilité: Lecture seule (Read only)
		Valeur par défaut: 0
Target_Join_Method	6	Type: Unsigned8
		Classification: Statique
		Accessibilité: Lecture seule (Read only)
		Valeur par défaut: 1
		Valeurs nommées: 0: clé symétrique; 1: clé asymétrique
Target_Security_Manager_EUI	7	Type: AdresseEUI64
		Classification: Statique
		Accessibilité: Lecture seule (Read only)
Target_System_Manager_Address	8	Type: IPv6Address
		Classification: Statique
		Accessibilité: Lecture seule (Read only)
		Plage valide: toutes les valeurs avec réglage sur bit le plus élevé
Target_Channel_List	9	Type: Array of Unsigned8
		Classification: Statique
		Accessibilité: Lecture seule (Read only)
Target_DL_Config	10	Type: OctetString (voir DL_Config_Info pour le format)
		Classification: Statique
		Accessibilité: Lecture seule (Read only)
Allow_Provisioning	11	Type: Boolean1
		Classification: Statique
		Accessibilité: Lecture/écriture
		Valeur par défaut: vrai
Allow_Default_Join	12	Type: Boolean1
		Classification: Statique
		Accessibilité: Lecture/écriture
		Valeur par défaut: vrai

[a] Lorsqu'un délai d'expiration de clé calculé donne la valeur 0xFFFF FFFF comme résultat, il est augmenté (circulairement) à la valeur modulo 0x0000 0000, ainsi la valeur 0xFFFF FFFF peut être utilisée pour désigner une clé qui n'expire jamais.

Quand il n'est pas égal à null, Device_Tag spécifie un Tag_Name affecté à l'appareil par un utilisateur. Cette valeur doit être écrite dans l'attribut Tag_Name du DMO (voir 6.2.8).

13.10.3 Méthode de l'objet service de configuration d'appareil

Plusieurs méthodes sont disponibles pour manipuler le DPSO. Les méthodes normalisées telles que read et write peuvent être utilisées pour les MIB (SMIB) scalaires ou structurées

dans leur intégralité. Les méthodes décrites dans le présent document sont utilisées pour manipuler des tables. Ces méthodes permettent l'accès à une rangée particulière d'une SMIB structurée en fonction d'un champ clé unique.

Il est supposé que les tables ont un champ clé unique, qui peut être soit un seul élément, soit la concaténation de plusieurs éléments. Le champ clé est supposé être le(s) (quelques) premier(s) élément(s) de la table ou la concaténation des quelques premiers éléments de la table.

Le Tableau 373 décrit les méthodes pour la manipulation des MIB structurées. Ces méthodes sont basées sur les modèles Read_Row, Write_Row, et Delete_Row définis en Annexe J.

Tableau 373 – Table de manipulation de matrice

Nom du type d'objet normalisé: Device provisioning service object (DPSO, objet service de configuration d'appareil)		
Identificateur du type d'objet normalisé: 106		
Nom de méthode	**ID de méthode**	**Description de la méthode**
Setrow_WhiteListTbl	1	Méthode pour établir (soit écrire, soit modifier) la valeur d'une seule rangée de la matrice de liste blanche. La méthode utilise le modèle de méthode Write_Row défini à l'Annexe J avec les arguments suivants: Attribute_ID:14 (White_List_Array) Index 1: 1 (Device_EUI)
Getrow_WhiteListTbl	2	Méthode pour obtenir la valeur d'une seule rangée de la matrice de liste blanche. La méthode utilise le modèle de méthode Read_Row défini à l'Annexe J avec les arguments suivants: Attribute_ID:14 (White_List_Array) Index 1: 1 (Device_EUI)
Deleterow_WhiteListTbl	3	Méthode pour supprimer la valeur d'une seule rangée de la matrice de liste blanche. La méthode utilise le modèle de méthode Delete_Row défini à l'Annexe J avec les arguments suivants: Attribute_ID:14 (White_List_Array) Index 1: 1 (Device_EUI)

13.10.4 Alertes de l'objet service de configuration d'appareil

Le Tableau 374 décrit une alerte pour indiquer une tentative de rattachement effectuée par un appareil qui n'est pas sur la liste blanche.

Tableau 374 – Alerte de DPSO pour indiquer le rattachement par un appareil ne figurant pas sur la WhiteList

Nom du type d'objet normalisé: Device provisioning service object (DPSO, objet service de configuration d'appareil)					
Identificateur du type d'objet normalisé: 106					
Description de l'alerte: Alerte pour indiquer la demande de rattachement par un appareil ne figurant pas sur la liste blanche					
Classe d'alertes (Enumerated: alarme ou événement)	**Catégorie d'alertes (Enumerated: diagnostic d'appareil, diagnostic de comm., sécurité ou processus)**	**Type d'alertes (Enumerated: en fonction de la catégorie d'alertes)**	**Priorité d'alertes (Enumerated: haut, moyen, faible, journal seulement)**	**Type de données de la valeur**	**Description de la valeur incluse avec l'alerte**
0 = événement	2 = sécurité	0 = Not_On_ Whitelist_Alert	6 = Moyen	Type: AdresseEUI64	Adresse EUI64 d'un appareil pas dans la liste blanche

Le Tableau 375 décrit une alerte pour indiquer qu'une capacité inadéquate est disponible pour qu'un appareil rejoigne le réseau.

Tableau 375 – Alerte de DPSO pour indiquer une capacité inadéquate de rattachement d'un appareil

Nom du type d'objet normalisé: Device provisioning service object (DPSO, objet service de configuration d'appareil)					
Identificateur du type d'objet normalisé: 106					
Description de l'alerte: Alerte pour indiquer une capacité inadéquate de rattachement d'un appareil					
Classe d'alertes (Enumerated: alarme ou événement)	**Catégorie d'alertes (Enumerated: diagnostic d'appareil, diagnostic de comm., sécurité ou processus)**	**Type d'alertes (Enumerated: en fonction de la catégorie d'alertes)**	**Priorité d'alertes (Enumerated: haut, moyen, faible, journal seulement)**	**Type de données de la valeur**	**Description de la valeur incluse avec l'alerte**
0 = événement	2 = sécurité	1 = Inadequate_Join_ Capability_Alert	6 = Moyen	Type: struct { reason: Unsigned8; rejectedDevice: EUI64Address}	Le champ reason fournit un code de diagnostic. Valeurs nommées: 1: mauvaise clé de rattachement; 2: clé de rattachement expirée; 3: authentification échouée Le champ rejectedDevice spécifie l'adresse EUI64 de l'appareil qui a tenté de s'attacher

13.10.5 Résumé des attributs qui peuvent être configurés

Ce qui suit est un résumé des attributs qui sont configurés par le PD afin qu'un nouvel appareil puisse rejoindre un réseau cible. Ces attributs peuvent être configurés en utilisant

des méthodes par liaison radio (OTA) ou des méthodes OOB. La liste d'attributs configurés comporte les éléments suivants. La liste complète est définie par le Tableau 368, le Tableau 371 et le Tableau 372.

- Informations relatives à la confiance:
  - clé de rattachement symétrique (K_join);
  - Adresse EUI64 du gestionnaire de sécurité;
  - méthode de rattachement au réseau.
- Informations relatives à un réseau:
  - network ID et bitmask;
  - Adresse IPv6 du gestionnaire de sécurité;
  - configuration de DL (contient les supertrames, TsTemplate de liaison, informations de voies, etc., exigé pour rejoindre le réseau cible).

En outre, les bits de configuration (attributs 6..10 du DPO), décrivant le comportement de l'appareil, peuvent être mis par l'appareil de configuration.

13.11 Fonctions de configuration (informative)

13.11.1 Généralités

L'interface et les procédures de configuration décrites dans le présent document ne décrivent pas une interface homme-machine (IHM). La présente norme ne spécifie pas une IHM spécifique, mais elle décrit comment de tels outils peuvent être conçus pour configurer des appareils conformes à la présente norme. Dans des opérations en installation, un utilisateur peut saisir des données de configuration et accepter ou rejeter des appareils souhaitant être configurés, en utilisant un appareil tenu à la main ou une interface en un certain emplacement central.

Les scénarios de configuration à l'Article 13 sont des exemples de configuration en utilisant des méthodes décrites par la présente norme.

13.11.2 Exemples de méthodes de configuration

13.11.2.1 Généralités

Des exemples sont débattus dans le présent document concernant la façon dont les objets et les procédures de gestion décrits peuvent être utilisés pour configurer un appareil. Ces exemples utilisent les méthodes de configuration suivantes:

- configuration par liaison radio utilisant des clés de rattachement préinstallées;
- configuration utilisant des mécanismes hors bande;
- configuration par liaison radio utilisant des certificats à clés asymétriques (PKI, par exemple);
- configuration par liaison radio utilisant des routeurs d'annonce à double rôle; et
- configuration de routeurs dorsaux.

13.11.2.2 Configuration par liaison radio utilisant des clés de rattachement préinstallées

Les étapes pour configurer un appareil avec des informations de confiance préinstallées peuvent notamment être:

a) L'appareil arrive au site de déploiement avec des clés préinstallées. Les clés sont des clés de rattachement symétriques.

b) Une WhiteList d'adresses d'appareil et de leurs clés symétriques correspondantes est installée dans le gestionnaire de sécurité du réseau-cible. Le mécanisme par lequel ces clés sont installées dans le gestionnaire de sécurité ne relève pas du domaine d'application de la présente norme. La WhiteList et les clés symétriques correspondantes peuvent être envoyées par courrier électronique sécurisé, envoyées sur des CD, remises à la main et saisies au clavier, ou livrées en utilisant n'importe quel autre outil approprié.

c) Le réseau-cible peut utiliser un sous-ensemble de fréquences autorisées et peut fonctionner dans le voisinage (à savoir dans la gamme d'interférence) de plusieurs réseaux distincts conformes à la présente norme. Dans ce cas, les informations relatives à un réseau peuvent être configurées dans l'appareil. Ces informations permettent à l'appareil de répondre à des annonces issues de réseaux-cibles seulement, et également de se mettre à l'écoute pour détecter des annonces aux fréquences correctes et au temps correct, diminuant ainsi les interférences et les temps de rattachement. Les informations relatives à un réseau peuvent être configurées en utilisant un PD par l'intermédiaire d'un mécanisme de communication hors bande ou par l'intermédiaire de mécanismes par liaison radio. Lorsqu'il n'est pas configuré avec des informations spécifiques de réseau cible, l'appareil peut essayer toutes les voies et tenter de rejoindre tous les réseaux dans son voisinage.

d) L'appareil se met à l'écoute pour détecter des annonces issues d'un routeur d'annonce dans le réseau-cible. Une fois qu'une annonce est entendue, l'appareil envoie une demande de rattachement au gestionnaire de système/gestionnaire de sécurité du réseau cible. (Le processus de rattachement de sous-réseau est décrit en 7.4.)

e) Le gestionnaire de système/gestionnaire de sécurité vérifie sa WhiteList et puis vérifie pour voir si la clé de rattachement de l'appareil concorde avec la clé de rattachement pour l'appareil fournie au gestionnaire de sécurité. Si les clés de rattachement concordent, le gestionnaire de sécurité fournit à l'appareil une clé principale qui est une clé secrète partagée entre le gestionnaire de sécurité et l'appareil. En outre, des clés T et des clés D sont fournies, et un contrat est établi avec le nouvel appareil pour parachever le processus de rattachement de sous-réseau.

13.11.2.3 Configuration utilisant des mécanismes hors bande

Les étapes pour configurer un appareil en utilisant un appareil de configuration OOB pourrait notamment être:

a) Un appareil frais arrive au site utilisateur. L'appareil a des réglages par défaut et n'a pas des clés préinstallées.

b) Un PD (par exemple, un appareil tenu à la main) obtient une liste de clés symétriques générées par le gestionnaire de sécurité/gestionnaire de système du réseau-cible. Les clés symétriques sont limitées dans le temps. Les clés peuvent être un éventail de clés ou des paires clé/adresse EUI64 spécifiques à un appareil, à la discrétion du gestionnaire de sécurité/du gestionnaire de système. Ces informations sont stockées dans le PD.

c) L'appareil tenu à la main, chargé avec les clés symétriques, est apporté à proximité du nouvel appareil ou relié au nouvel appareil et il est établi une connexion OOB entre l'appareil tenu à la main et le DBP. L'appareil tenu à la main utilise alors l'interface de communication OOB pour peupler les attributs du DPO dans le nouvel appareil. La communication OOB peut se produire en infrarouge, par connexion physique, par communication de champ proche ou d'autres moyens.

d) L'appareil est maintenant prêt à rejoindre le réseau-cible et se met à l'écoute pour détecter des annonces issues du réseau-cible. L'appareil répond aux annonces en envoyant une demande de rattachement, par l'intermédiaire du routeur d'annonce, au gestionnaire de système cible. Le gestionnaire de sécurité vérifie sa WhiteList, le cas échéant. Le gestionnaire de sécurité vérifie également la validité de la clé de rattachement et vérifie que la clé n'a pas expiré. Si la clé est valide, le gestionnaire de sécurité accepte la demande de rattachement et fournit à l'appareil une clé principale qui est une clé secrète partagée entre le gestionnaire de sécurité et l'appareil. En outre, des clés T et des clés D sont également fournies, et un contrat est établi avec le nouvel appareil pour parachever le processus de rattachement de sous-réseau.

13.11.2.4 Configuration par liaison radio utilisant des certificats d'infrastructure à clés asymétriques

Les étapes pour configurer un appareil qui a des informations de confiance préinstallées pourraient notamment être:

a) L'appareil arrive au site de déploiement avec un module de sécurité installé. Le module contient un certificat signé en usine et une paire de clés publique/privée. Une autorité de certification (CA) a signé la clé d'émetteur de l'usine.

b) Une WhiteList d'adresses d'appareil et une liste de clés asymétriques des autorités de certification sont installées dans le gestionnaire de sécurité du réseau-cible. Le mécanisme par lequel ces clés sont installées dans le gestionnaire de sécurité ne relève pas du domaine d'application de la présente norme. La WhiteList peut être envoyée par courrier électronique sécurisé, envoyée sur des CD, remise à la main et saisie au clavier, ou livrée en utilisant n'importe quel autre outil approprié.

c) Le réseau-cible peut utiliser un sous-ensemble de voies autorisées, parce qu'il peut fonctionner dans le voisinage (à savoir au sein de la gamme d'interférence) de plusieurs réseaux. Dans un tel cas, des informations relatives à un réseau peuvent être configurées à l'appareil, permettant à l'appareil de répondre aux annonces seulement issues des réseaux-cibles et de se mettre à l'écoute pour détecter des annonces sur les voies prévues, aux instants appropriés, diminuant ainsi les interférences et augmentant les fréquences de rattachement. Si l'appareil n'est pas configuré avec des informations spécifiques de réseau cible, l'appareil peut essayer toutes les voies et essayer de rejoindre tous les réseaux dans son voisinage. Les informations relatives à un réseau sont configurées en utilisant un appareil de configuration par l'intermédiaire d'un mécanisme de communication OOB ou par l'intermédiaire de mécanismes par liaison radio.

d) L'appareil se met maintenant à l'écoute pour détecter les annonces issues du routeur d'annonce dans le réseau-cible. Une fois qu'une annonce est entendue, l'appareil partage ses certificats avec le gestionnaire de sécurité. Avec le certificat public de la CA, le gestionnaire de sécurité décode le certificat d'appareil et vérifie qu'il est valide. La procédure implique également un mécanisme de défi-réponse de la part du gestionnaire de système/gestionnaire de sécurité pour confirmer l'identité de l'appareil se joignant. Le gestionnaire de sécurité vérifie sa WhiteList pour confirmer que l'appareil est censé rejoindre le réseau. L'étape de confirmation peut impliquer un menu contextuel sur une GUI du gestionnaire de sécurité pour une confirmation manuelle par un utilisateur. Une fois confirmé, le gestionnaire de sécurité peut émettre des clés T pour l'appareil si l'appareil souhaite rejoindre immédiatement le réseau. En variante, le gestionnaire de sécurité peut émettre des clés de rattachement symétrique pour que l'appareil rejoigne le réseau à un moment ultérieur. Dans un cas comme dans l'autre, les clés émises sont retournées à l'appareil, chiffrées avec la clé publique de l'appareil.

13.11.2.5 Configuration par liaison radio utilisant des routeurs d'annonce à double rôle

Les étapes pour configurer un appareil par liaison radio en utilisant un routeur d'annonce à double rôle pourraient notamment être:

a) L'appareil arrive avec des réglages d'usine par défaut en un site utilisateur. Le site utilisateur requiert de très faibles niveaux de sécurité.

b) Certains des routeurs d'annonce au site utilisateur ont un double rôle et fonctionnent comme des appareils de configuration. En utilisant la clé de rattachement symétrique ouverte (K_join = K_open), le routeur d'annonce à double rôle (à savoir le côté PD logique du routeur d'annonce à double rôle) forme un mini-réseau avec le nouvel appareil et fournit au nouvel appareil les valeurs de réglage réseau et la clé de rattachement pour le réseau cible. Ces valeurs de réglage, y compris les clés, sont envoyées en clair par liaison radio. L'appareil à double rôle peut aussi informer le gestionnaire de sécurité du réseau cible de mettre à jour sa liste blanche en ajoutant l'appareil qui vient juste d'être configuré. L'appareil de configuration à double rôle peut être opérationnel dans un endroit où l'utilisateur a une confiance certaine que l'émission de clés de rattachement par liaison radio présente un faible risque. (Cette étape pose le risque similaire à celui qui est pris en liant les ouvre-portes de garage à des télécommandes.)

c) Le DPO du nouvel appareil a maintenant les informations relatives à la confiance et les informations relatives à un réseau pour rejoindre le réseau-cible. Il peut utiliser les routeurs d'annonce (soit le même routeur d'annonce à double rôle qui l'a configuré, ou un autre routeur d'annonce du réseau cible si l'appareil a été déplacé) et envoie une demande de rattachement au gestionnaire de système du réseau cible.

d) Le gestionnaire de système du réseau-cible accepte la demande de rattachement et fournit un contrat au nouvel appareil.

13.11.2.6 Configuration de routeurs dorsaux

Les étapes pour configurer des appareils dorsaux pourraient notamment être:

a) Un appareil frais arrive au site utilisateur. L'appareil a des réglages par défaut et n'a pas de clés préinstallées.

b) Un PD (par exemple, un appareil tenu à la main ou un appareil connecté par l'intermédiaire de l'interface de dorsale au DBP) obtient une liste de clés symétriques générées par le gestionnaire de sécurité/gestionnaire de système du réseau-cible. Les clés symétriques sont limitées dans le temps. Les clés peuvent être un éventail de clés ou des paires clé/adresse EUI64 spécifiques à un appareil, à la discrétion du gestionnaire de sécurité/du gestionnaire de système. Ces informations sont stockées dans le DPSO du PD.

c) Le PD, chargé avec les clés symétriques, est apporté à proximité du nouvel appareil ou relié au nouvel appareil et il est établi une connexion OOB (qui peut, dans ce cas, être la dorsale) entre l'appareil tenu à la main et le DBP. Le PD utilise alors l'interface de communication OOB (plus probablement l'interface de dorsale) pour peupler les attributs du DPO dans le nouvel appareil.

d) L'appareil est maintenant prêt à rejoindre le réseau-cible. Cependant, à la différence d'un appareil de terrain, un appareil dorsal peut ne pas avoir une interface de DL. Par exemple, l'appareil peut être une passerelle résidant sur la dorsale. En variante, l'appareil dorsal peut être le premier routeur d'annonce connecté au réseau. Par exemple, l'appareil peut être un routeur dorsal avec la fonctionnalité de routeur d'annonce sur l'interface de couche physique IEEE 802.15.4:2011. Cependant, l'appareil a besoin d'être configuré sur la dorsale et pas par l'intermédiaire de la PhL, car, dans ce cas, il n'y a pas de routeurs d'annonce qui peuvent transmettre sa demande de rattachement au gestionnaire de système.

Pour parler au gestionnaire de système sur la dorsale sans l'aide d'un routeur d'annonce, le routeur dorsal envoie une demande de rattachement au gestionnaire de système directement sur la dorsale; l'appareil dorsal peut former l'en-tête de réseau nécessaire pour envoyer ce message. Il peut le faire parce qu'il a été configuré avec l'adresse IPv6 du gestionnaire de système (DPO.Target_System_Manager_Address). La suite de la procédure au niveau du gestionnaire de système est la même que celle en 13.11.2.3.

Annexe A
(informative)

Couche d'utilisateur/profils d'application

A.1 Vue d'ensemble

L'Annexe A décrit ce que signifient les termes "couche d'utilisateur" et "profil d'application" et elle décrit également comment ces termes se rapportent l'un à l'autre et à la présente norme.

A.2 Couche d'utilisateur

La couche d'utilisateur correspond au terme souvent appliqué à une huitième couche inexistante placée au sommet du modèle de mise en réseau d'ordinateurs à sept couches de l'OSI. L'intention de la couche d'utilisateur est d'accomplir des fonctions spécifiques à un but qui ne sont pas liées à des communications réseau. En ce qui concerne l'automation industrielle, le terme "couche d'utilisateur" est parfois appliqué pour décrire le matériel et/ou le logiciel qui n'est pas relié à la communication réseau, tel qu'un capteur de terrain ou un bloc fonctionnel de contrôle de processus. Il est possible qu'une telle couche d'utilisateur ait des informations qui doivent être communiquées sur un autre réseau qui se conforme à l'ISO/IEC 7498, le modèle de référence de base de l'OSI.

La communication réseau pour prendre en charge la fonction de couche d'utilisateur est initiée par la couche d'utilisateur en utilisant les méthodes et les protocoles définis par la 7ème couche du modèle OSI, qui s'appelle Application Layer (AL).

La présente norme vise à prendre en charge une diversité de fonctions de l'industrie d'automation qui ne sont pas directement liées à la communication réseau. À ce titre, elle définit une pile de communication d'usage général compatible avec le modèle de mise en réseau d'ordinateurs de l'OSI et inclut la définition des services normalisés de l'AL.

La présente norme définit également des objets normalisés extensibles génériques, qui peuvent être utilisés par des applications d'automation industrielle. La présente norme permet la spécialisation de ces objets normalisés, tout comme la définition à la fois de nouveaux objets normalisés spécifiques à une industrie et d'objets définis par un fournisseur. La définition d'objets normalisés spécifiques à une industrie ne relève pas du domaine d'application de la présente norme; elle est laissée à chaque organisation industrielle qui promeut l'utilisation de la présente norme pour leur industrie. La présente norme ne limite pas la portée de la fonctionnalité de couche d'utilisateur par rapport à un besoin quelconque de communication non liée à un réseau.

NOTE L'ISA 100 Wireless Compliance Institute (WCI) est un exemple d'une telle organisation pour les industries d'automation de processus.

A.3 Profil d'application

Un profil d'application définit les propriétés spécifiques à une application devant être mises en œuvre d'une manière qui promeut l'interopérabilité parmi les entités en communication. Un profil d'application peut également définir des politiques de mise en œuvre, et peut suggérer des lignes directives de mise en œuvre. Toute couche d'utilisateur au sein d'un appareil peut mettre en œuvre un ou plusieurs profils d'application.

Certaines propriétés spécifiques à un profil d'application peuvent être obligatoires pour toutes les instances des applications conformes au profil d'application particulier. D'autres propriétés spécifiques à un profil d'application peuvent être des propriétés de pratique courante qui sont des options de construction. Toutes ces propriétés sont représentées sous la forme d'attributs

d'objet, si bien que la communication de leurs valeurs peut se produire par l'utilisation des services de base de couche d'application selon la présente norme.

La portée d'un profil d'application est souvent délibérément limitée, afin de promouvoir une plus grande adoption et une plus grande utilisation du profil d'application particulier. Un exemple d'un tel profil d'application limité est un profil d'application pour des capteurs de température.

Dans un système faiblement couplé, la liaison des appareils qui prennent en charge des profils d'application avec des applications de système hôte qui utilisent ces profils est habituellement accomplie par l'intermédiaire de l'utilisation d'un fichier de caractérisation d'appareil fourni par le fournisseur de l'appareil, dont le contenu est souvent basé sur une technologie descriptive normalisée.

Un exemple d'une norme qui peut être utilisée pour décrire le contenu de profil est l'IEC 61804-3, qui peut être utilisée par les fournisseurs d'appareils pour l'industrie d'automation industrielle pour créer le fichier qui peut être utilisé avec les outils appropriés d'accompagnement de système hôte, permettant à l'hôte de représenter des fonctions d'appareil, des paramètres (attributs) et leurs dépendances, les représentations graphiques appropriées à la représentation de données, ainsi que des interactions prises en charge avec d'autres appareils.

Un appareil peut mettre en œuvre un profil d'application ou un ensemble de profils et peut utiliser les méthodes d'AL natives et les protocoles de la présente norme pour communiquer sur des réseaux sans fil conformes à la présente norme.

Sachant que la présente norme est censée prendre en charge une diversité de fonctions d'industrie d'automation industrielle qui ne sont pas liées à la communication réseau, la présente norme ne définit pas ou ne limite pas la définition ou l'utilisation des profils d'application, des langues ou des fichiers qui représentent de tels appareils, ou des outils utilisés pour représenter de tels appareils. La définition de profils normalisés spécifiques à une industrie ne relève donc pas du domaine d'application de la présente norme. Elle est, au lieu de cela, déléguée aux entreprises qui promeuvent l'utilisation de la présente norme dans une industrie d'automation particulière.

NOTE Des mécanismes ISO et IEC existent pour proposer de tels profils d'application spécifiques à une industrie.

Annexe B
(normative)

Profils de rôles de communications

B.1 Vue d'ensemble

B.1.1 Généralités

Un profil de rôles est défini comme représentant les capacités de base, y compris toutes les valeurs de réglages et toutes les configurations, qui sont exigées d'un appareil pour accomplir correctement le rôle en question. Les rôles sont définis en 5.2.6.2, mais ils sont énumérés ici pour référence comme étant gestionnaire de système, gestionnaire de sécurité, routeur dorsal, routeur, entrée-sortie (E/S), passerelle, source de temps de système, et appareil de configuration.

L'Annexe B fournit le pro forma de profil de rôle pour la conformité à la présente norme.

B.1.2 Objet

Le profil de rôle définira les capacités d'appareil, telles que des valeurs de réglage et des configurations, nécessaires pour remplir chaque rôle spécifique défini en 5.2.6.2. Le but en est d'assurer que les appareils conformes à la présente norme, y compris l'Annexe B, peuvent être interopérables, selon le cas, dans le domaine couvert par le profil de rôle.

B.1.3 Taille de système

Les capacités exigées d'un appareil pour mettre en œuvre un rôle peuvent dépendre du nombre d'appareils dans le système prévu. La taille minimale de système est définie à l'Article 5, mais il n'y a aucune taille maximale de système. Pour permettre aux exigences de l'Annexe B de servir une large plage de tailles de système, ces exigences dépendant de la taille de système doivent utiliser une formule pour spécifier les capacités minimales. Pour les besoins de l'Annexe B, le nombre d'appareils du système est appelé NSD (number of system devices).

B.1.4 Abréviations et symboles spéciaux

Les abréviations et les symboles utilisés comprennent:

- Notations pour le statut d'exigence:
  - O: obligatoire;
  - F: facultative;
  - O.n: facultative, mais la prise en charge d'au moins l'un des groupes d'options étiquetées "O.n" est exigée;
  - N/A: non applicable;
  - X: interdit.
- Élément: Conditionnelle, statut dépendant de la prise en charge marquée pour l'élément.

Par exemple, un statut de FD1:O.1 et FD2:O.1 indique que le statut est facultatif, mais au moins une des caractéristiques décrites dans FD1 et FD2 est exigée.

B.1.5 Profils de rôles

Le Tableau B.1 décrit les couches de protocoles et les exigences relatives aux supports pour tous les profils de rôles. Si un appareil est déclaré prendre en charge plus d'un rôle, cet appareil doit remplir les capacités minimales pour chaque rôle déclaré.

Tableau B.1 – Rôles des appareils de couche de protocoles

Numéro d'élément	Rôle de l'appareil	Etat					Référence	Support		
		Couches de protocole			Moyenne					
		AL	TL	NL	Type A	Squelette		N/A	Oui	Non
DR1	E/S	M	M	M	M	N/A	5.2.6.6			
DR2	Routeur	M	M	M	M	N/A	5.2.6.7			
DR3	Routeur dorsal	M	M	M	M DR4: O DR7: O	M	5.2.6.9 5.2.6.9 5.2.6.9			
DR4	Passerelle	M	M	M	DR2:O.1	DR3:O.1	5.2.6.10			
DR5	Source de temps système	N/A	N/A	N/A	N/A	N/A	5.2.6.13			
DR6	Configuration	M	M	M	M	N/A	5.2.6.8			
DR7	Gestionnaire de système	M	M	M	DR2:O.2	DR3:O.2	5.2.6.11			
DR8	Gestionnaire de sécurité	N/A	N/A	N/A	N/A	N/A	5.2.6.12			

B.2 Système

Le protocole de WISN prend en charge la capacité de mettre à niveau des appareils par liaison radio, conformément au Tableau B.2.

Tableau B.2 – Mises à niveau par liaison radio

Numéro d'élément	Types de rôles affectés	Référence	Etat	Support		
				N/A	Oui	Non
OTAR1	E/S		M			
OTAR2	Routeur		M			
OTAR3	Routeur dorsal		N/A			
OTAR4	Passerelle		N/A			
OTAR5	Gestionnaire de système		N/A			
OTAR6	Appareil de configuration		O			

B.3 Gestionnaire de système

Le gestionnaire de système alloue la capacité des appareils pour communiquer en générant, distribuant et maintenant des contrats qui définissent les ressources nécessaires pour ce besoin de communication. Sachant que chaque appareil doit stocker ses contrats, la capacité d'un appareil pour le stockage de contrats est critique. Alors que les capacités nécessaires des appareils E/S, routeurs et routeurs dorsaux dépendent du nombre d'objets d'application au sein des appareils en question, la passerelle et le gestionnaire de système dépendent du

nombre d'appareils dans le système, défini dans l'Annexe B comme étant NSD. Le NSD n'inclut pas le gestionnaire de système dans son compte d'appareils.

Les contrats exigent des sessions de communication pour la communication, établies par le gestionnaire de sécurité conjointement avec le gestionnaire de système. Plusieurs contrats, communiquant vers les mêmes points d'extrémité, peuvent partager une seule session. Les capacités minimales décrites ici supposent que chaque session est appariée à un seul contrat, en admettant que plus de contrats peuvent être nécessaires selon la nature des applications de l'appareil.

B.4 Gestionnaire de sécurité

Le gestionnaire de sécurité établit des sessions entre les processus d'application. Par exemple, lorsqu'un appareil rejoint le réseau, il a besoin d'une session DMAP-SMAP. Le nombre de sessions qu'un appareil mettant en œuvre un rôle doit être capable de maintenir est défini dans le Tableau B.3. Le nombre de sessions prises en charge par un gestionnaire de système dépend de NSD. Le nombre de clés prises en charge par une passerelle dépend du nombre de connexions Gateway-UAP que la passerelle est conçue pour prendre en charge, appelé GUC (gateway-UAP connection) dans le Tableau B.3.

Un appareil E/S est présumé requérir la capacité pour prendre en charge les sessions suivante:

- Une session entre le DMAP de l'appareil et le SMAP, établie lorsque l'appareil rejoint le réseau.
- Une session entre l'UAP de l'appareil et un premier appareil tel qu'une passerelle.
- Une session entre le DMAP de l'appareil et le premier appareil, pour rapporter des alertes de processus.
- Une session entre l'UAP de l'appareil et l'UP du second appareil, telle que dans le cas d'une communication peer-to-peer.

Tableau B.3 – Profils de prise en charge de sessions

Numéro d'élément	Types de rôles affectés	Nombre minimal de sessions prise en charge	Commentaires	Etat	Support		
					N/A	Oui	Non
NCS1	E/S	4	DMAP-SMAP UAP-Gateway DMAP-Gateway UAP-Peer	M			
NCS2	Routeur	1	DMAP-SMAP	M			
NCS3	Routeur dorsal	1	DMAP-SMAP	M			
NCS4	Passerelle	(2 x GUC) + 1	DMAP-SMAP GUC x (Gateway-UAP) GUC x (Gateway-DMAP)	M			
NCS5	Gestionnaire de système	NSD	NSD x (SMAP-DMAP)	M			

Le gestionnaire de sécurité assigne les clés de sécurité qui sont exigées pour la communication entre appareils. Le nombre de clés qu'un appareil mettant en œuvre un rôle doit être capable de maintenir est défini dans le Tableau B.4. Le nombre de clés prises en charge pour un appareil dépend du nombre de sessions prises en charge, avec les capacités minimales montrées dans le Tableau B.3. En outre, chaque appareil a besoin de la capacité

pour une clé de rattachement, une clé principale, et une clé D si une DL est incluse sur l'appareil. Des comptes de clé ont besoin d'être doublés, parce que toutes les clés excepté la clé de rattachement peuvent être dans le processus de changement.

Tableau B.4 – Profils de base

Numéro d'élément	Types de rôles affectés	Nombre minimal de clés prise en charge	Commentaires	Référence	Etat	Support		
						N/A	Oui	Non
NKS1	E/S	1+((NCS1+2)×2)		7.2.2	M			
NKS2	Routeur	1+((NCS2+2) ×2)		7.2.2	M			
NKS3	Routeur dorsal	1+((NCS3+2) ×2)		7.2.2	M			
NKS4	Passerelle	1+((NCS4+1) ×2)	Ajouter 2 si la passerelle a un DL	7.2.2	M			
NKS5	Gestionnaire de système	(NCS5+1) ×2	Ajouter 2 si SM a un DL					
NKS7	Gestionnaire de sécurité	–N/A		7.2.2	N/A			

B.5 Physical layer, couche physique

Sachant que la PhL cite les spécifications issues de l'IEEE 802.15.4, les capacités de rôle pour la PhL font référence à l'IEEE 802.15.4.

Le Tableau B.5 décrit les rôles de couche physique.

Tableau B.5 – Rôles de PhL

Numéro d'élément	Descriptif de l'élément		Référence IEEE 802.15.4:2011	Etat	Support		
					N/A	Oui	Non
PLR1	E/S	L'appareil est un appareil à fonction réduite	5.1	O.1			
		L'appareil est un appareil à fonction complète	5.1	O.1			
PLR2	Routeur	L'appareil est un appareil à fonction complète	5.1	M			
PLR3	Routeur dorsal	L'appareil est un appareil à fonction complète	5.1	M			
PLR4	Appareil de configuration	L'appareil est un appareil à fonction complète	5.1	M			
O.1: au moins une option doit être sélectionnée.							

B.6 Data-link layer, couche de liaison de données

B.6.1 Généralités

La DL affecte quatre profils de rôles, conformément au Tableau B.6.

Tableau B.6 – DL exigée pour rôles énumérées

Numéro d'élément	Types de rôles	Référence	Etat	Support		
				N/A	Oui	Non
DLR1	E/S	5.2.6.6	M			
DLR2	Routeur	5.2.6.7	M			
DLR3	Routeur dorsal	5.2.6.9	M			
DLR4	Configuration	5.2.6.8	M			

B.6.2 Profils de rôles

B.6.2.1 Généralités

Un profil de rôles de DL décrit un ensemble de capacités minimales qui doivent être prises en charge par chaque appareil conforme qui met en œuvre le moyen de terrain de Type A. Par exemple, un appareil remplissant le rôle de routeur doit prendre en charge 8 voisins. Si un appareil satisfait à toutes les autres exigences d'un routeur, mais prend en charge seulement 4 voisins, il n'est pas conforme dans son rôle de routeur. Un appareil peut dépasser n'importe lesquelles des exigences de son rôle, tant que toutes les exigences minimales des rôles sont respectées.

La DL est configurée par l'intermédiaire de valeurs de réglages des attributs de l'objet de gestion de DL (DLMO), et les divers rôles sont décrits comme plages de valeurs de réglages de DLMO qu'un appareil peut prendre en charge.

B.6.2.2 Attributs d'objet de gestion de DL

Un niveau d'appareil de prise en charge pour une capacité peut être exprimé en rapport avec un ensemble d'attributs de DLMO et d'éléments de ces attributs. Chaque attribut et/ou élément dont la prise en charge varie par rôle est inclus.

Si un certain nombre ou une plage de nombres est énuméré(e), un appareil remplissant ce rôle doit prendre en charge le nombre en question. Si un seul nombre est énuméré, il doit être interprété comme étant une valeur minimale, sauf indication contraire. Par exemple, si un appareil doit prendre en charge 3 voisins, alors il peut prendre en charge 4 voisins, mais il est non conforme s'il prend en charge seulement 2 voisins. Un appareil E/S peut être capable de routage même s'il n'est pas entièrement conforme au rôle de routeur; par conséquent, certaines capacités relatives au routage sont montrées comme étant facultatives (non interdites) pour un appareil E/S.

Le Tableau B.7 décrit des attributs de DLMO simples avec un seul élément. (Les autres tableaux de l'Annexe B traitent des attributs de DLMO contenant plusieurs éléments.)

Tableau B.7 – Profils de rôles: Attributs généraux de DLMO

Attribut	Etat			Commentaires	Support		
	E/S	Routeur	BBR		N/A	Oui	Non
ActScanHostFract	O	M	M	Un appareil qui n'est pas un appareil d'alimentation n'aura pas nécessairement l'énergie pour agir comme un hôte de balayage actif pendant une période de temps prolongée. Voir Tableau B.8			
AdvJoinInfo AdvSuperframe	O	M	M	Tous les routeurs et routeurs dorsaux peuvent être configurés pour envoyer des annonces			
TaiTime TaiAdjust	M	M	M	La DL n'est pas nécessairement la source de TaiTime pour un appareil particulier et il existe des cas où la DL d'un appareil ne pourrait pas être impliquée dans la propagation de temps comme une source ou une destinataire. Par exemple, un BBR pourrait rester synchronisé en temps par l'intermédiaire d'un mécanisme de dorsale, et ne pas être impliqué dans la propagation de temps de DL			
ClockTimeout	M	M	M	Un BBR peut être configuré comme un destinataire d'horloge, mais cela n'est pas censé être typique			

Le Tableau B.8 décrit des profils de rôles de base pour l'attribut dlmo.Device_Capability. Les éléments d'appareils non mentionnés dans l'Annexe B doivent être pris en charge comme cela est décrit à l'Article 9.

Tableau B.8 – Profils de rôles: dlmo.Device_Capability

Elément	Etat			Notes de bas de tableau	Support		
	E/S	Routeur	BBR		N/A	Oui	Non
QueueCapacity	0	10	20	a			
ClockStability	100	10	10	b			
DLRoles	0000 xxx1	0000 xx1x	0000 x11x	c			
AdvRate	0 (X)	6	6	d			
ListenRate	0 (X)	36	36	e			
TransmitRate	0 (X)	30	60	f			

a Le gestionnaire de système configure la file d'attente de DL seulement dans la mesure où l'appareil transmet des messages pour le compte d'autres appareils. La file d'attente de DL dans un BBR est un sujet interne d'appareil pour les graphes qui commencent ou finissent dans la DL de l'appareil.

b Les valeurs de ClockStability, comme multiples de 1×10^{-6} sont des valeurs maximales permises sur tout intervalle continu de 30 s dans des conditions de fonctionnement industrielles. Alors que les appareils E/S à bas coût peuvent avoir des horloges avec une stabilité à court terme de seulement 100×10^{-6}, il convient que les appareils E/S industriel en général aient une meilleure stabilité. La présente norme a été conçue en admettant des horloges de 25×10^{-6} dans les appareils d'E/S ou mieux et il est anticipé que la plupart des profils d'application seront contraints en conséquence.

c Les bits indiquent tous les rôles de DL qui sont pris en charge par l'appareil. Noter que BBR doit agir comme un routeur, comme dans le cas d'une messagerie peer-to-peer dans un sous-réseau D.

d Tous les appareils desservant des rôles de routeur et de routeur dorsal doivent avoir les ressources suffisantes pour émettre une annonce toutes les 10 secondes (6 DPDU par minute), en moyenne. Voir 9.1.17.

e Tous les appareils desservant des rôles de routeur et de dorsal doivent avoir les ressources suffisantes pour exploiter leurs récepteurs pendant 36 secondes par heure (1 %) en moyenne. Un BBR alimenté par secteur sera normalement capable de faire fonctionner son récepteur de manière continue, mais certaines classes de BBR (telles que les ponts sans fil) pourraient être contraintes en énergie.

f Tous les appareils desservant des rôles de routeur et de dorsale doivent avoir les ressources suffisantes pour émettre le nombre indiqué de DSDU par minute. Voir 9.1.17.

Le Tableau B.9 décrit des profils de rôles de base pour l'attribut dlmo.Ch. Les éléments d'appareils non mentionnés dans l'Annexe B doivent être pris en charge comme cela est décrit à l'Article 9.

Tableau B.9 – Profils de rôles: dlmo.Ch (channel-hopping)

Elément	Etat			Commentaires	Support		
	E/S	Routeur	BBR		N/A	Oui	Non
Capacity (métadonnées)	10	10	10	Cinq modèles de séquences de voie par défaut, numérotés 1..5, sont définis par la norme. Un appareil peut être configuré avec un maximum de 5 séquences de sauts de voie supplémentaires			
MaxRowID (métadonnées)	127	127	127	Un octet			

Le Tableau B.10 décrit des profils de rôles de base pour l'attribut dlmo.TsTemplate. Les éléments d'appareils non mentionnés dans l'Annexe B doivent être pris en charge comme cela est décrit à l'Article 9.

Tableau B.10 – Profils de rôles: dlmo.TsTemplate

Elément	Etat			Commentaires	Support		
	E/S	Routeur	BBR		N/A	Oui	Non
Capacity (métadonnées)	8	10	10	Trois modèles d'intervalle de temps par défaut, numérotés 1..3, sont définis par la présente norme. Ils sont inclus dans la capacité			
MaxRowID (métadonnées)	127	127	127	Un octet			

Le Tableau B.11 décrit des profils de rôles de base pour l'attribut dlmo.Neighbor. Les éléments d'appareils non mentionnés dans l'Annexe B doivent être pris en charge comme cela est décrit à l'Article 9.

Tableau B.11 – Profils de rôles: dlmo.Neighbor

Elément	Etat			Commentaires	Support		
	E/S	Routeur	BBR		N/A	Oui	Non
Capacity (métadonnées)	2	8	32	Un E/S doit prendre en charge au moins deux voisins afin de pouvoir maintenir deux routes DL actives pour la création des rapports. Un routeur ajoute de la capacité supplémentaire pour prendre en charge le routage pour le compte de voisins			
MaxRowID (métadonnées)	2^{15}	2^{15}	2^{15}	Adresse unicast 6LoWPAN limitée à 2^{15}			
GroupCode	O	M	M	GroupCode permet à des liaisons d'être utilisées pour plusieurs voisins			
ExtendGraph	O	O	O	L'extension automatique de graphes est exigée pour tous les appareils. La prise en charge du champ ExtendGraph est une option de construction qui fournit un degré plus fin de contrôle sur les extensions de graphes			

Le Tableau B.12 décrit des profils de rôles de base pour l'attribut dlmo.Diagnostic. Les éléments d'appareils non mentionnés dans l'Annexe B doivent être pris en charge comme cela est décrit à l'Article 9.

Tableau B.12 – Profils de rôles: dlmo.NeighborDiag

Elément	Etat			Commentaires	Support		
	E/S	Routeur	BBR		N/A	Oui	Non
Capacity (métadonnées)	2 × 15 + 1 × 9	3 × 15 + 2 × 9	3 × 15 + 2 × 9	La capacité de diagnostic (métadonnées) est mesurée en octets. Les diagnostics de résumés, dans le Tableau 188, concernent 15 octets de stockage dans le pire des cas. Le stockage et l'émission réels peuvent être plus compacts. Les diagnostics de résumé sont censés être maintenus du côté "publication" d'une liaison donnée, pour recueillir les diagnostics à partir du sens où circule plus de trafic. Les diagnostics de résumé incluent un diagnostic d'horloge de base (ClockSigma). Des diagnostics d'horloge plus détaillés (Tableau 190) concernent 9 octets de stockage dans le pire des cas. Un diagnostic d'horloge de résumé est fourni avec le diagnostic général. La capacité est fournie de recueillir ces diagnostic d'horloge détaillés selon les besoins			
MaxRowID (métadonnées)	2^{15}	2^{15}	2^{15}	Adresse unicast 6LoWPAN limitée à 2^{15}			

Le Tableau B.13 décrit des profils de rôles de base pour l'attribut dlmo.Superframe. Les éléments d'appareils non mentionnés dans l'Annexe B doivent être pris en charge comme cela est décrit à l'Article 9.

Tableau B.13 – Profils de rôles: dlmo.Superframe

Elément	Etat			Commentaires	Support		
	E/S	Routeur	BBR		N/A	Oui	Non
Capacity (métadonnées)	3	5	10	Les supertrames par défaut pour la découverte de l'appareil de configuration sont incluses dans ce compte			
MaxRowID (métadonnées)	127	127	127	Un octet			
AlwaysHop	O	O	O	La prise en charge de cette caractéristique est une option de construction			

Le Tableau B.14 décrit des profils de rôles de base pour l'attribut dlmo.Graph. Les éléments d'appareils non mentionnés dans l'Annexe B doivent être pris en charge comme cela est décrit à l'Article 9.

Tableau B.14 – Profils de rôles: dlmo.Graph

Elément	Etat			Commentaires	Support		
	E/S	Routeur	BBR		N/A	Oui	Non
Capacity (métadonnées)	2	8	16				
MaxRowID (métadonnées)	127	127	127	Un octet			

Le Tableau B.15 décrit des profils de rôles de base pour l'attribut dlmo.Link. Les éléments d'appareils non mentionnés dans l'Annexe B doivent être pris en charge comme cela est décrit à l'Article 9.

Tableau B.15 – Profils de rôles: dlmo.Link

Elément	Etat			Commentaires	Support		
	E/S	Routeur	BBR		N/A	Oui	Non
Capacity (métadonnées)	9	15	30	Les liaisons par défaut pour la découverte de l'appareil de configuration sont incluses dans ce compte			
MaxRowID (métadonnées)	127	127	127	Un octet			
Discovery	0, 3	0, 1, 2, 3	0, 1, 2	Discovery se réfère aux bits 3/2 dans le Tableau 182. Un gestionnaire de système peut être configuré pour découvrir des voisins capables de routage par l'intermédiaire de la recherche d'annonces par balayage actif ou passif			
JoinResponse	O	M	M				
NeighborType=2	O	M	M	La prise en charge de groupes de voisins est obligatoire pour des appareils de routage			

Le Tableau B.16 décrit des profils de rôles de base pour l'attribut dlmo.Route. Les éléments d'appareils non mentionnés dans l'Annexe B doivent être pris en charge comme cela est décrit à l'Article 9.

Tableau B.16 – Profils de rôles: dlmo.Route

Elément	Etat			Commentaires	Support		
	E/S	Routeur	BBR		N/A	Oui	Non
Capacity (métadonnées)	3	1	64	L'appareil E/S a la capacité pour le routage vers le gestionnaire de système, un premier appareil et un second appareil. Le routeur a besoin seulement d'un chemin vers le gestionnaire de système. BBR a besoin au moins d'un chemin (consultation de chemin sortant) pour chaque appareil dans sa sphère d'influence, même si ces chemins sont identiques les uns aux autres			
MaxRowID (métadonnées)	127	127	127	Un octet			

Le Tableau B.17 décrit des profils de rôles de base pour l'attribut dlmo.Queue_Priority. Les éléments d'appareils non mentionnés dans l'Annexe B doivent être pris en charge comme cela est décrit à l'Article 9.

Tableau B.17 – Profils de rôles: dlmo.Queue_Priority

Elément	Etat			Commentaires	Support		
	E/S	Routeur	BBR		N/A	Oui	Non
Capacity (métadonnées)	O	2	2				
MaxRowID (métadonnées)	127	127	127	Un octet			

B.7 Couche réseau

Le Tableau B.18 décrit des profils de rôles pour des tailles de table de routage.

Tableau B.18 – Taille de table de routage

Numéro d'élément	Types de rôles affectés	Nombre minimal d'entrées prise en charge	Commentaires	Référence	Etat	Support		
						N/A	Oui	Non
RTS1	E/S	0			M			
RTS2	Routeur	0			M			
RTS3	Routeur dorsal	15			M			

LeTableau B.19 décrit des profils de rôles pour des tailles de table d'adresses.

Tableau B.19 – Taille de table d'adresses

Numéro d'élément	Types de rôles affectés	Nombre minimal d'entrées prise en charge	Commentaires	Référence	Etat	Support		
						N/A	Oui	Non
ATS1	E/S	4			M			
ATS2	Routeur	3			M			
ATS3	Routeur dorsal	15			M			

B.8 Couche transport

Le Tableau B.20 décrit des profils de rôles pour des tailles de support de ports.

Tableau B.20 – Taille de support de ports

Numéro d'élément	Types de rôles affectés	Nombre minimal d'entrées prise en charge	Commentaires	Référence	Etat	Support		
						N/A	Oui	Non
PSS1	E/S	2			M			
PSS2	Routeur	1			M			
PSS3	Routeur dorsal	1			M			

B.9 Couche d'application

Le Tableau B.21 décrit le nombre minimal d'AP par rôle.

Tableau B.21 – AP

Numéro d'élément	Types de rôles affectés	Nombre minimal d'AP pris en charge	Commentaires	Référence	Etat	Support		
						N/A	Oui	Non
UAPO1	E/S	2		Article 6, 12.17	M			
UAP02	Routeur	1		Article 6	M			
UAP03	Routeur dorsal	1		Article 6	M			
UAP04	Passerelle	2		Article 6, Annexe U	M			
NOTE Le nombre maximal d'objets contenus pris en charge inclut l'UAPMO.								

B.10 Configuration

Le Tableau B.22 fournit les appareils à profil de rôles mettant en œuvre les rôles d'E/S, de routeur, de passerelle, ou les rôles de routeur dorsal, tous les appareils ayant un moyen de terrain de Type A.

Tableau B.22 – Profils de rôles: E/S, routeurs, passerelles, et routeurs dorsaux

Numéro d'élément	Caractéristique intrinsèque	Référence	Etat	Plage	Commentaires	Support		
						N/A	Oui	Non
DBPR-1	Rattachement à un réseau de configuration à l'aide de K_global	13.6	M		Voir 7.2.2.2			
DBPR-2	Rattachement à un réseau de configuration à l'aide de K_open	13.6	O		Valeur par défaut de K_join = K_open. Désactivé une fois que S est écrasé en écriture. Activé à nouveau seulement si appareil réinitialisé aux valeurs d'usine par défaut			

B.11 Passerelle (informative)

Le Tableau B.23 fournit le profil de rôles pour une passerelle.

Tableau B.23 – Profil de rôles: Passerelle

Numéro d'élément	Caractéristique intrinsèque	Référence	Etat	Commentaires	Support		
					N/A	Oui	Non
GWRP1	Accès natif	U.3.1.5	O.1	Permet l'accès de service natif seulement			
GWRP2	Mécanisme de tunnellisation interopérable	U.3.1.5	O.1	Autorise l'accès tunnellisé seulement			

Le Tableau B.24 fournit le profil notionnel de rôles pour un accès natif de passerelle.

Tableau B.24 – Profil de rôles: Accès natif à la passerelle

Numéro d'élément	Caractéristique intrinsèque	Référence	Etat	Commentaires	Support		
					N/A	Oui	Non
GWRP1.1	IFO min pris en charge	U.3.1.5	1				
GWRP1.2	Comportement de messages tamponnés	U.3.4		Constant, static, dynamic, non-cacheable.			
GWRP1.3	Appareils min	Tableau 373	NSD	NSD ≥ 5			
GWRP1.4	Locations min	Tableau 373	2 x NSD – 3	NSD ≥ 5			

Le Tableau B.25 fournit le profil de rôle notionnel pour un mécanisme de tunnellisation interopérable de passerelle.

Tableau B.25 – Profil de rôles: Mécanisme de tunnellisation interopérable de passerelle

Numéro d'élément	Caractéristique intrinsèque	Référence	Etat	Commentaires	Support		
					N/A	Oui	Non
GWRP2.1	TUN min pris en charge	U.3.1.5	GD x AD + 1	GD ≥ 1 AD ≥ 5			
GWRP2.1.1	Prend en charge un protocole étranger	U.3.1.5	Annexe O				
GWRP2.1.2	Tunnellisation à 2 parties	U.3.1.5					
GWRP2.1.3	Objets TUN avec attributs Array of Tunnel endpoints avec plusieurs éléments d'adresse	U.3.1.5	1				
GWRP2.1.3.1	Nombre d'éléments dans un objet TUN avec plusieurs éléments d'adresse	U.3.1.5	A	A ≥ 5			
GWRP2.2	Appareils min	Tableau 373	A	A ≥ 5			
GWRP2.3	Locations min	Tableau 373	2 x A	A ≥ 5			

Annexe C
(informative)

Informations de référence

C.1 Besoins industriels

Les besoins en sans fil pour des applications industrielles sont sensiblement différents de ceux exigés pour des applications résidentielles, commerciales, ou militaires. Ces différences découlent du classement industriel unique des priorités des caractéristiques telles que le coût d'appareil, le coût de système, le coût de cycle de vie, la fiabilité, la maintenabilité, la cohérence, la robustesse, l'extensibilité, la sécurité, la coexistence, les restrictions dues à la réglementation, l'interopérabilité et (dans les domaines adéquats) l'interfonctionnabilité ou l'interopérabilité.

Les membres de comité ISA 100 ont recueilli et analysé plus de 500 cas d'utilisation pour définir plus complètement les besoins en communication sans fil du secteur industriel. Les principales conclusions de cet effort étaient:

- Occasion: La détection sans fil inexistante est une occasion pour les utilisateurs finaux, les fournisseurs et les normes émergentes.
- Interopérable: Sachant que de multiples moyens instrumentaux de fournisseurs dominent l'environnement industriel, il convient que les normes du sans fil soient de haute valeur.
- Interopérable: Il convient que les appareils qui ciblent le même domaine d'application vaste (par ex: contrôle des processus ou gestion d'actifs) soient interopérables en ce qui concerne les fonctionnalités de base nécessaires pour une action coopérative dans ce domaine d'application.
- Intégration: De multiples chemins de communication entre les appareils sont nécessaires, particulièrement vers les instruments de système de commande distribué (DCS).
- Applications: Les applications telles que la surveillance/l'alerte sont du plus grand intérêt immédiat, car elles constituent la plus grande utilisation potentielle des appareils sans fil.
- Fiabilité et sécurité: Facteurs critiques pour les normes émergentes et les fournisseurs.
- Puissance: Les espérances de vie de batterie varient en raison de l'application, de l'environnement, des contraintes de coût, etc. Certains appareils auront l'alimentation en énergie par le secteur, alors que d'autres seront alimentés par des batteries ou récupéreront l'énergie dans l'environnement.

C.2 Classes d'utilisation

C.2.1 Généralités

Alors qu'il existe de nombreuses techniques qui peuvent être utilisées pour catégoriser les besoins en communications des applications industrielles, la présente norme utilise des classes basées sur l'utilisation. L'analyse des profils d'utilisation prévue des communications sans fil industrielles entre appareils a conduit au partitionnement de telles communications en six classes. Ces classes sont résumées dans le Tableau C.1.

Tableau C.1 – Classes d'utilisation

Sécurité	Classe 0: Action d'urgence	Toujours critique
Commande	Classe 1: Commande de réglementation à boucle fermée	Souvent critique
	Classe 2: Commande de supervision à boucle fermée	Habituellement non critique
	Classe 3: Commande à boucle ouverte	Humain dans la boucle
Contrôle	Classe 4: Alerte	Conséquence opérationnelle de court terme (maintenance événementielle, par exemple)
	Classe 5: Journalisation et téléchargement descendant/ téléchargement montant	Aucune conséquence opérationnelle immédiate (par exemple, recueil d'antécédents, séquence d'événements, maintenance préventive)
NOTE Les niveaux de lot 3 et 4 pourraient être de classe 2, de classe 1 ou même de classe 0, selon la fonction. Les niveaux de lot sont définis dans l'IEC 61512-1, où L3 = unité et L4 = cellule de processus.		

C.2.2 Exemples de classes

- Classe 0: Action d'urgence (toujours critique)

  On peut citer les exemples suivants:

  – interverrouillage de sécurité;
  – arrêt d'urgence;
  – contrôle incendie automatique.

- Classe 1: Commande de régulation en circuit fermé (souvent critique)

  On peut citer les exemples suivants:

  – commande directe d'actionneurs primaires (par exemple, disponibilité de la connexion appareil de terrain à hôte à la demande d'au moins 99,99 %, avec coupures de liaison > 500 ms intolérables, avec des fréquences de demande de 0,2 Hz ou plus);
  – boucles haute fréquence en cascade.

- Classe 2: Commande de supervision à boucle fermée (habituellement non critique)

  On peut citer les exemples suivants:

  – boucles basse fréquence en cascade;
  – commandes à plusieurs variables;
  – optimiseurs.

- Classe 3: Commande à boucle ouverte (Homme dans la boucle)

  On peut citer les exemples suivants:

  – l'opérateur initie manuellement une flamme et observe la flamme;
  – le garde ouvre à distance un portail de sécurité;
  – l'opérateur accomplit un réajustement manuel de pompe/vanne.

- Classe 4: Alerte – Conséquence opérationnelle de court terme

  On peut citer les exemples suivants:

  – maintenance événementielle;
  – temps marginal de palier conduit à envoyer un technicien sur le terrain;
  – le voyant batterie faible pour un appareil conduit à envoyer un technicien pour changer la batterie;
  – suivi de biens.

- Classe 5: Journalisation – données/messages sans conséquence opérationnelle immédiate

  On peut citer les exemples suivants:

  – recueil d'antécédents;

  – cycle de maintenance préventive;

  – chargement de séquence d'événements (SOE).

NOTE La SOE utilise une communication sans pertes, telle qu'un transfert de fichier, plutôt qu'une communication minutée telle qu'utilisée par la messagerie de commande.

C.2.3 Autres alarmes de téléchargement montant et de téléchargement descendant (action humaine ou automatisée)

Des exemples d'alarmes sont:

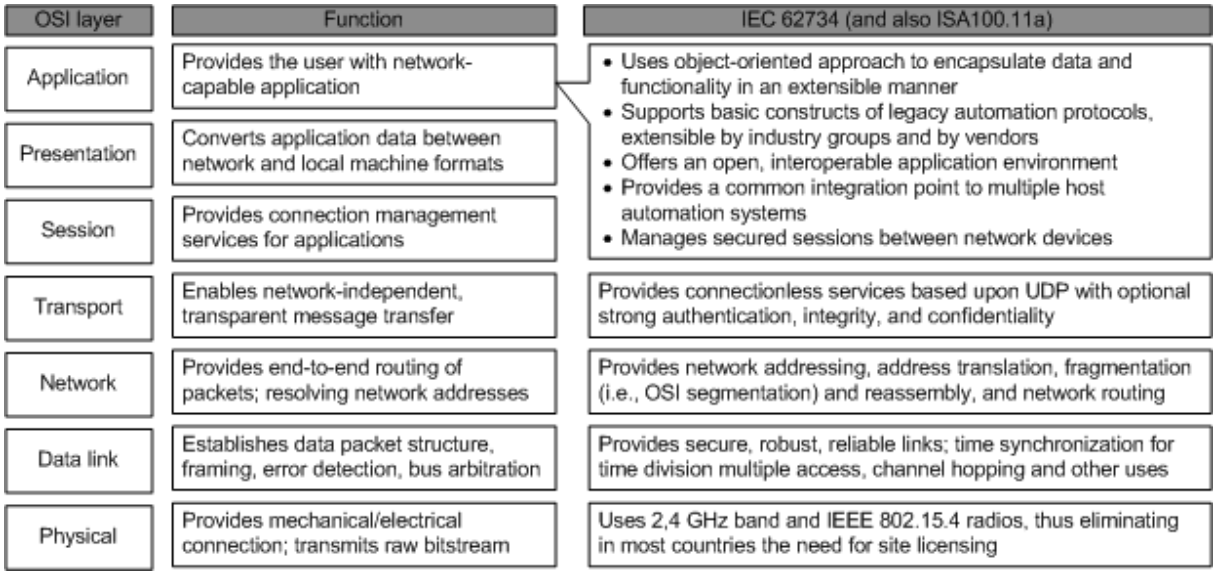
- Classe 0: détecteur de fuite pour rayonnement ou gaz mortellement toxique, réponse automatisée (par exemple, réponse de confinement automatisée).
- Classe 1: condition de processus à impact élevé, réponse automatisée (par exemple, arrêt automatisé d'une réaction).
- Classe 2: réponse automatisée à l'état du processus (par exemple, diversion automatisée de flux).
- Classe 3: état de processus avec réponse opérationnelle initiée manuellement (par exemple, décider, oui ou non, de dévier le flux vers un réacteur parallèle).
- Classe 4: état du matériel avec réponse de maintenance à courte échelle de temps (par exemple, envoyer un technicien sur le terrain).
- Classe 5: état du matériel avec action de maintenant à longue échelle de temps (par exemple, commander des pièces de rechange).

C.3 Modèle de référence de base d'interconnexion des systèmes ouverts (OSI)

C.3.1 Vue d'ensemble

La présente norme définit la suite de protocoles du réseau sans fil. Une suite de protocole est une mise en œuvre particulière de logiciel d'une suite de protocoles de mise en réseau. Dans une mise en œuvre pratique, les suites de protocoles sont souvent divisées en couches telles que celles définies dans le Modèle de référence de base de l'Interconnexion des systèmes ouverts ISO/IEC 7498-1. Le format dans la présente norme est basé sur ce modèle de référence (voir la Figure C.1), mettant en œuvre cinq des sept couches du modèle de référence de base.

NOTE Il est utile de réaliser qu'il s'agit d'un modèle virtuel, qui n'impose donc aucune exigence sur les mises en œuvre, ou même de spécifications.



Légende

Anglais	Français
OSI layer	Couche OSI
Application	Application
Presentation	Présentation
Session	Session
Transport	Transport
Network	Réseau
Data link	Liaison de données
Physical	Physique
Function	Fonction
Provides the user with network-capable application	Fournit à l'utilisateur des applications à capacité de réseau
Converts application data between network and local machine formats	Convertit les données d'application entre les formats de réseau et de machines locales
Provides connection management services for applications	Fournit des services de gestion de connexion pour les applications
Enables network-independent, transparent message transfer	Permet un transfert de messages transparent et indépendant vis-à-vis du réseau
Provides end-to-end routing of packets; resolving network addresses	Fournit un routage de bout en bout de paquets, résolvant les adresses réseau
Establishes data packet structure, framing, error detection, bus arbitration	Etablit la structure des paquets de données, le verrouillage de trame, la détection d'erreur et les contrôles d'accès aux bus
Provides mechanichal/electrical connnection; transmits raw bitstream	Fournit la connexion mécanique/électrique; émet le train de bits brut
IEC 62734 (and also ISA100.11a)	IEC 62734 (et également ISA100.11a)

Anglais	Français
Uses object-oriented approach to encapsulate data and functionality in an extensible manner Supports basic constructs of legacy automation protocols, extensible by industry groups and by vendors Offers an open, interoperable application environment Provides a common integration point to multiple host automation systems Manages secured sessions between network devices	Utilise une approche orientée objet pour encapsuler les données et la fonctionnalité de manière extensible Prend en charge les constructions de base des protocoles d'automatisation existants, extensibles par groupes d'industries et par fournisseurs Offre un environnement d'application ouvert et interopérable Gère les sessions sécurisées entre les appareils réseau
Provides connectionless services based upon UDP with optional strong authentication, integrity, and confidentiality	Fournit des services sans connexion basés sur UDP avec authentification, intégrité et confidentialité fortes facultatives
Provides network addressing, address translation, fragmentation (i.e., OSI segmentation) and reassembly, and network routing	Fournit l'adressage réseau, la traduction des adresses et la fragmentation (c'est-à-dire la segmentation OSI) ainsi que le réassemblage et le routage réseau
Provides secure, robust, reliable links; time synchronization for time division multiple access, channel hopping and other uses	Fournit des liaisons sécurisées, robustes et fiables; synchronisation temporelle pour l'accès multiple et le saut de voies par répartition dans le temps, ainsi que d'autres utilisations
Uses 2,4 GHz band, IEEE 802.15.4 radios, thus eliminating in most countries the need for site licensing	Utilise des radios de bande 2,4 GHz, IEEE 802.15.4, ce qui élimine le besoin de licence des sites dans la plupart des pays

Figure C.1 – Modèle de référence de base de l'OSI

La couche supérieure, application (AL), du Modèle de référence de base de la présente norme fournit la fonctionnalité locale pour un ou plusieurs UAP associés.

Les quatre couches, transport (TL), réseau (NL), liaison de données (DL), et physique (PhL), sont consacrées à la communication de données. Chacune a les capacités de multiplexer et de démultiplexer, et de diviser et de fusionner les flux d'informations issus des couches adjacentes. Autrement dit, les relations de messagerie entre une entité d'AL et une entité de TL, ou entre une entité de TL et une entité de NL, ou entre une entité de NL et une entité de DL, ou entre une entité de DL et une entité de PhL, peuvent ne pas être biunivoques (1:1).

Ces couches inférieures ont aussi les capacités suivantes:

- ordonnancer des unités de données de service (SDU) pour maintenir l'ordre de la présentation originale;
- exécuter une ou plusieurs des opérations ci-dessous
  - segmenter ou réassembler des SDU dans des unités de données de protocole (PDU),
  - bloquer ou débloquer des SDU dans des unités de données de protocole (PDU) et
  - concaténer ou séparer des PDU,

  afin qu'elles soient dimensionnées de manière plus appropriée pour les capacités de transport de la couche inférieure;
- diviser des PDU parmi plusieurs chemins de couche inférieure ou recombiner ces PDU à leur réception avant de les transférer sur une route de couche supérieure; et
- acquitter la réception de PDU comme une forme de contrôle d'erreur.

C.3.2 Couche d'application

L'AL est la couche qui s'interface directement aux UAP (et les inclut conceptuellement), gérant des communications avec d'autres UAP sous la supervision de l'UAP de gestion

locale. Un UAP peut accomplir une fonction individuelle ou n'importe quelle combinaison de fonctions. Les UAP peuvent être utilisés, par exemple, pour:

- traiter les matériels d'entrée et/ou de sortie;
- distribuer des communications vers un jeu d'UAP corésident au sein d'un appareil (fonction proxy);
- prendre en charge la tunnellisation d'un protocole non natif (par exemple, hérité d'un système de commande) compatible avec l'environnement réseau de la présente norme; et/ou
- accomplir une fonction de calcul.

L'AL est habituellement composée d'un ou plusieurs UAP qui partagent des éléments de services communs.

Les tâches principales d'une entité d'AL permettent de fournir:

- une place dans l'architecture de la présente norme pour les UAP;
- les moyens par lesquels les UAP gèrent les communications avec des UAP pour d'autres appareils par l'intermédiaire de la suite de protocoles, y compris:
  - identification des partenaires de communication prévus (par exemple, par le nom, par l'adresse, par la description, etc.),
  - accord sur la sécurité (par exemple, authentification, intégrité des données),
  - détermination de la qualité de service acceptable (par exemple, priorité, fenêtres de temps pour la messagerie de commande, acceptabilité de la livraison de message dans le désordre, acceptabilité de la livraison de message en des incréments partiels, etc.),
  - accord sur la responsabilité pour la correction d'erreur,
  - identification des syntaxes abstraites et
  - synchronisation des UAP coopérateurs;
- les moyens par lesquels les UAP peuvent informer l'entité d'application associée des exigences de ressource nécessaires, y compris celles applicables pour le placement en tampon des messages:
  - tailles attendues et maximales des messages et
  - nombre de salves maximal prévu de l'émission et de la réception des messages ou combien de messages peuvent être envoyés ou arriver en une courte quantité de temps en comparaison à la périodicité moyenne des messages; et
- toute fonction nécessaire de communication qui ne soit pas déjà accomplie par les couches inférieures.

C.3.3 Couche transport

La TL est la couche la plus élevée à laquelle les applications en communication sont adressables. Les tâches principales d'une entité de TL sont:

- fournir l'adressage des UAP par l'intermédiaire de la sélection d'une entité d'AL associée spécifique;
- établir les chemins de messagerie de bout en bout allant d'un UAP à un ou plusieurs autres UAP par l'intermédiaire de leurs entités d'AL associées, où ces processus sont habituellement situés dans des appareils distincts;
- acheminer et réguler le flux de messages entre ou parmi ces UAP; et
- mettre fin à ces chemins de messagerie s'il y a lieu.

C.3.4 Couche réseau

La NL est la couche la plus élevée à laquelle les appareils en communication sont adressables. Il s'agit de la plus basse couche avec une portée plus que locale, qui transmet des messages entre un groupe d'entités et d'autres, ou rejette les messages. Les tâches principales d'une entité de NL sont:

- fournir l'adressage, à l'échelle d'un réseau, des appareils;
- relayer des messages (les NPDU) entre les entités (par exemple, un routeur) par l'intermédiaire des sous-réseaux D, habituellement en changeant les adresses d'entité de DL source et destination associées à des enveloppes de message (les DPDU) dans le processus, ou rejeter les NPDU; et
- fournir la segmentation et le réassemblage de messages, selon le cas, pour correspondre aux capacités des sous-réseaux D sur lesquels des messages sont transmis.

NOTE La NL est la couche OSI où ont lieu l'adressage et le routage d'appareils de point d'extrémité. Les relais de couche inférieure peuvent transmettre des messages au sein d'un seul domaine d'adressage sans modification de message, mais ils sont incapables de réadresser des messages ou de couvrir des domaines d'adressage. Les adresses d'appareil à l'échelle d'un réseau sont les adresses IPv6.

C.3.5 Couche de liaison de données

La DL est la plus basse couche centrée sur les informations, qui coordonne les entités de PhL interagissantes et fournit la messagerie de bas niveau de base parmi des entités de DL. Les tâches principales d'une entité de DL sont:

- fournir l'adressage, local à une liaison, des entités DL homologues;
- acheminer des messages (les DPDU) d'une entité de DL donnée vers toutes les autres dont les entités de PhL sont des correspondantes (par exemple, à toutes les entités de PhL de la liaison locale), ou rejeter les DPDU;
- gérer l'utilisation de la PhL;
- fournir l'adressage de message de bas niveau, la temporisation des messages et les contrôles d'intégrité des messages;
- fournir la détection de bas niveau de la perte de message et la récupération de perte de message (par exemple, acquittement immédiat; répéter la tentative si aucun acquittement); et
- facultativement, relayer les DPDU entre les entités de DL (par exemple, un pont).

NOTE La DL est la couche OSI qui gère et compense les caractéristiques spécifiques de la technologie de communications physiques choisie. Elle fournit seulement l'adressage local, et transmet des messages au sein du domaine d'adressage local sans réadressage. Elle ne modifie pas les adresses de message. Les adresses DL16 ont seulement une portée locale et, donc, il est possible que les mêmes adresses DL16 soient dupliquées dans d'autres liaisons locales.

C.3.6 Couche physique

La PhL est la couche la plus basse du modèle OSI et la seule couche qui traite de la physique du monde réel. Toutes autres couches traitent d'informations abstraites, représentées en dernier ressort sous forme de bits; la PhL est concernée par les signaux physiques (parfois appelés bauds ou puces). Les tâches principales d'une entité de PhLE sont:

- coder les bits, séparément ou en groupes de plusieurs bits, dans les signaux physiques;
- acheminer ces signaux d'un emplacement physique vers un autre;
- décoder ces signaux en groupes d'un seul bit ou de plusieurs bits, possiblement avec correction d'erreur;
- prendre des instructions auprès du DLE associé en ce qui concerne l'installation de voie physique, l'adressage de récepteur physique et d'autres aspects de voie et de codage de communications;

- transporter au DLE localement associé les informations relatives à l'état de l'entité de PhLE, de la voie et du dernier jeu de signaux reçus; et
- facultativement, relayer les PhPDU entre les entités de PhLE (par exemple, un répétiteur).

Annexe D
(normative)

Valeurs de configuration par défaut

D.1 Généralités

L'Annexe D résume les réglages par défaut pour la configuration.

D.2 Gestion de systèmes

Le Tableau D.1 énumère les valeurs par défaut de la configuration de la gestion de système.

Tableau D.1 – Valeurs par défaut de la configuration de la gestion de système

Nom	Valeur par défaut initiale	Référence
Confirmation_Timeout_Device_Diagnostics	10	Tableau 7
Alerts_Disable_Device_Diagnostics	0	Tableau 7
Confirmation_Timeout_Comm_Diagnostics	10	Tableau 7
Alerts_Disable_Comm_Diagnostics	0	Tableau 7
Confirmation_Timeout_Security	10	Tableau 7
Alerts_Disable_Security	0	Tableau 7
Confirmation_Timeout_Process	10	Tableau 7
Alerts_Disable_Process	0	Tableau 7
Comm_Diagnostics_Alarm_Recovery_AlertDescriptor	Valeur par défaut: [faux, 3]	Tableau 7
Security_Alarm_Recovery_AlertDescriptor	Valeur par défaut: [faux, 3]	Tableau 7
Device_Diagnostics_Alarm_Recovery_AlertDescriptor	Valeur par défaut: [faux, 3]	Tableau 7
Process_Alarm_Recovery_AlertDescriptor	Valeur par défaut: [faux, 3]	Tableau 7
DL_Alias_16_Bit	0	Tableau 10
Network_Address_128_Bit	0	Tableau 10
Device_Power_Status_Check_AlertDescriptor	Valeur par défaut: [faux, 8]	Tableau 10
DMAP_State	1	Tableau 10
Join_Command	0	Tableau 10
Static_Revision_Level	0	Tableau 10
Restart_Count	0	Tableau 10
Uptime	0	Tableau 10
TAI_Time	0	Tableau 10
Comm_SW_Major_Version	0	Tableau 10
Comm_SW_Minor_Version	0	Tableau 10
System_Manager_128_Bit_Address	0	Tableau 10
System_Manager_EUI64	0	Tableau 10
System_Manager_DL_Alias_16_Bit	0	Tableau 10
Contract_Request_Timeout	30	Tableau 10
Max_ClientServer_Retries	3	Tableau 10
Max_Retry_Timeout_Interval	30	Tableau 10
DMAP_Objects_Count	1	Tableau 10
Warm_Restart_Attempts_Timeout	60	Tableau 10
Current_UTC_Adjustment	35	Tableau 25
Next_UTC_Adjustment_Time	Voir Tableau 25	Tableau 25
Next_UTC_Adjustment	35	Tableau 25

D.3 Sécurité

Le Tableau D.2 énumère les valeurs par défaut de la configuration de la sécurité.

Tableau D.2 – Valeurs par défaut de la configuration de la sécurité

Nom	Valeur par défaut initiale	Référence
Security_Level	1	Tableau 87
Protocol_Version	1	Tableau 92
DL_Security_Level	1	Tableau 92
Transport_Security_Level	1	Tableau 92
Join_Timeout	60 s	Tableau 92
MPDU_MIC_Failure_Limit	5	Tableau 92
MPDU_MIC_Failure_Time_Unit	60 s	Tableau 92
TPDU_MIC_Failure_Limit	5	Tableau 92
TPDU_MIC_Failure_Time_Unit	5	Tableau 92
DSMO_KEY_Failure_Limit	1	Tableau 92
DSMO_KEY_Failure_Time_Unit	1	Tableau 92
Security_MPDU_Fail_Rate_Exceeded_AlertDescriptor	[faux, 6]	Tableau 92
Security_TPDU_Fail_Rate_Exceeded_AlertDescriptor	[faux, 6]	Tableau 92
Security_Key_Update_Fail_Rate_Exceeded_AlertDescriptor	[faux, 6]	Tableau 92
pduMaxAge	510	Tableau 92
SoftLifeTime	50	Tableau 93
DSMO alert type 0 = Security_MPDU_Fail_Rate_Exceeded	0	Tableau 97
DSMO alert type 1 = Security_TPDU_Fail_Rate_Exceeded	0	Tableau 97
DSMO alert type 2 = Security_Key_Update_Fail_Rate_Exceeded	0	Tableau 97

D.4 Data-link layer, couche de liaison de données

Le Tableau D.3 énumère les valeurs par défaut de la configuration DLE.

Tableau D.3 – Valeurs par défaut de la configuration DLE

Nom	Valeur par défaut initiale	Référence
ActScanHostFract	0	Tableau 141
AdvJoinInfo	Null	Tableau 141
AdvSuperframe	0	Tableau 141
SubnetID	0	Tableau 141
SolicTemplate	Null	Tableau 141
AdvFilter	Voir 9.4.2.20	Tableau 141
SolicFilter	Voir 9.4.2.20	Tableau 141
TaiAdjust	Null	Tableau 141
MaxBackoffExp	5	Tableau 141
MaxDsduSize	96	Tableau 141
MaxLifetime	120 (30 s)	Tableau 141
NackBackoffDur	60 (15 s)	Tableau 141
LinkPriorityXmit	8	Tableau 141
LinkPriorityRcv	0	Tableau 141
EnergyDesign	Voir 9.4.2.22	Tableau 141
DeviceCapability	Voir 9.4.2.23	Tableau 141
IdleChannels	0	Tableau 141
ClockExpire	Voir 9.4.2.1	Tableau 141
ClockStale	45	Tableau 141
RadioSilence	600	Tableau 141
RadioSleep	0	Tableau 141
RadioTransmitPower	Voir 9.4.2.1	Tableau 141
CountryCode	0x3C00	Tableau 141
Candidates	Null	Tableau 141
DiscoveryAlert	60	Tableau 141
SmoothFactors	Voir Tableau 153	Tableau 141
QueuePriority	N=0	Tableau 141
Ch	Voir 9.4.3.2	Tableau 141
TsTemplate	Voir 9.4.3.3	Tableau 141
Neighbor (voisin)	Vide	Tableau 141
Superframe	Vide	Tableau 141
Courbe	Vide	Tableau 141
Liaison	Vide	Tableau 141
Route	Vide	Tableau 141
NeighborDiag	Vide	Tableau 141
ChannelDiag	Voir 9.4.2.27	Tableau 141
Paramètres de modèle de récepteur de transaction	Voir Tableau 165	Tableau 165
Paramètres de modèle d'initiateur de transaction	Voir Tableau 166	Tableau 166
Modèle de récepteur de transaction pour analyse des paramètres	Voir Tableau 167	Tableau 167

D.5 Couche Réseau

Le Tableau D.4 énumère les valeurs par défaut de la configuration NLE.

Tableau D.4 – Valeurs par défaut de la configuration NLE

Nom	Valeur par défaut initiale	Référence
Enable_Default_Route	faux	Tableau 206
Max_NSDU_size	70	Tableau 206
Frag_Reassembly_Timeout	60	Tableau 206
Frag_Datagram_Tag	aléatoire uniforme	Tableau 206
DroppedNPDUAlertDescriptor	[vrai, 7]	Tableau 206
Source_Address*	0	Tableau 207
Destination_Address	0	Tableau 207
Contract_Priority	00	Tableau 207
Include_Contract_Flag	faux	Tableau 207
NWK_HopLimit	64	Tableau 208
Outgoing_Interface	0	Tableau 208

D.6 Couche Transport

Le Tableau D.5 énumère les valeurs par défaut de la configuration TLE.

Tableau D.5 – Valeurs par défaut de la configuration TLE

Nom	Valeur par défaut initiale	Référence
MaxNbOfPorts	15	Tableau 229
TPDUin	0	Tableau 229
TPDUinRejected	0	Tableau 229
TSDUout	0	Tableau 229
TSDUin	0	Tableau 229
TSDUinRejected	0	Tableau 229
TPDUout	0	Tableau 229
IllegalUseOfPortAlertDescriptor	[true, 8] -- medium	Tableau 229
TPDUonUnregisteredPortAlertDescriptor	[true, 4] -- low	Tableau 229
TPDUoutOfSecurityPoliciesAlertDescriptor	[true, 2] -- journal	Tableau 229

D.7 Application layer, couche d'application

Le Tableau D.6 énumère les valeurs par défaut de la configuration ALE.

Tableau D.6 – Valeurs par défaut de la configuration ALE

Nom	Valeur par défaut initiale	Référence
ObjectIdentifier	0	Tableau 240
UAP_ID	0=N/A	Tableau 240
UAP_TL_Port	0=N/A	Tableau 240
Etat	Active	Tableau 240
Commande	0=None	Tableau 240
MaxRetries	3	Tableau 240
Nombre de correspondants de communication non programmés	0=N/A	Tableau 240
Nombre d'objets dans l'UAP y compris cet UAPMO	1	Tableau 240
Static_Revision_Level	0	Tableau 240
Catégories	0	Tableau 243
Erreurs	0	Tableau 243
Etat	0=Idle	Tableau 246
MaxBlockSize	1..(MaxNPDUsize + Max TL header size – max(sizeof (additional coding of AL UploadData service request), additional coding of sizeof(AL DownloadData service response))	Tableau 246
MaxDownloadSize	0	Tableau 246
MaxUploadSize	0	Tableau 246
DownloadPrepTime	0	Tableau 246
DownloadActivationTime	0	Tableau 246
UploadPrepTime	0	Tableau 246
UploadProcessingTime	0	Tableau 246
DownloadProcessingTime	0	Tableau 246
CutoverTime	0	Tableau 246
LastBlockDownloaded	0	Tableau 246
LastBlockUploaded	0	Tableau 246
ErrorCode	0 =(noError)	Tableau 246
Revision	0	Tableau 256
CommunicationEndpoint	L'élément valide de point d'extrémité de connexion configuré indique non configuré (c'est-à-dire que le point d'extrémité n'est pas valide)	Tableau 256
MaximumItemsPublishable	Local matter	Tableau 256
NumberItemsPublishing	0	Tableau 256
Array of ObjectAttributeIndexAndSize	La taille de l'élément est 0	Tableau 256
Concentrator ContentRevision	0	Tableau 258
CommunicationEndpoint	L'élément valide de point d'extrémité de connexion configuré indique non configuré (c'est-à-dire que le point d'extrémité n'est pas valide)	Tableau 258
MaximumItemsSubscribing	Local matter	Tableau 258
NumItemsSubscribing	0	Tableau 258
Array of ObjectAttributeIndexAndSize	La taille de l'élément est 0	Tableau 258
Protocole	Initiative locale (spécifique au protocole)	Tableau 260
Status (Configuration status)	0	Tableau 260
Période (période de publication de données)	0	Tableau 260
Max_Peer_Tunnels	0	Tableau 260
Num_Peer_Tunnels	0	Tableau 260
ObjectIdentifier	7	Tableau 283
MalformedAPDUsAdvise	faux	Tableau 283
TimeIntervalForCountingMalformedAPDUs	0	Tableau 283
MalformedAPDUsThreshold	0	Tableau 283
MalformedAPDUAlertDescriptor	[vrai, 7]	Tableau 283
PV	NaN	Tableau 287
Mode	OOS	Tableau 287
Echelle	Les valeurs des unités d'étude pour 0 % et pour 100 % indiquent TOUTES LES DEUX 0	Tableau 287
OP	NaN	Tableau 290
Mode	OOS	Tableau 290
Readback	NaN	Tableau 290
Echelle	Les valeurs des unités d'étude pour 0 % et pour 100 % indiquent TOUTES LES DEUX 0	Tableau 290
PV_B	0	Tableau 293
Mode	Lecture seule pour le mode réel; le mode cible, le mode autorisé et le mode normal ont tous un accès en lecture/écriture	Tableau 293
OP_B	0	Tableau 296
Mode	Lecture seule pour le mode réel; le mode cible, le mode autorisé et le mode normal ont tous un accès en lecture/écriture	Tableau 296
Readback_B	0	Tableau 296

Nom	Valeur par défaut initiale	Référence
Cible	OOS	Tableau 302
Actual	OOS	Tableau 302
Autorisé	OOS	Tableau 302
Valeur normale	OOS	Tableau 302
Unités d'étude à 100 %	0	Tableau 304
Unités d'étude à 0 %	0	Tableau 304
Decimal point location	0	Tableau 304

D.8 Configuration

Le Tableau D.7 énumère les valeurs par défaut de la configuration de configuration.

Tableau D.7 – Valeurs par défaut de la configuration de configuration

Nom	Valeur par défaut initiale	Référence
Default_NWK_ID	0x0001	Tableau 368
Default_SYM_Join_Key	K_global	Tableau 368
Open_SYM_Join_Key	K_open	Tableau 368
Default_Channel_List	0x7FFF	Tableau 368
Join_Method_Capability	00	Tableau 368
Allow_Provisioning	vrai (1)	Tableau 368
Allow_Over_The_Air_Provisioning	vrai (1)	Tableau 368
Allow_OOB_Provisioning	vrai (1)	Tableau 368
Allow_Reset_to_Factory_defaults	vrai (1)	Tableau 368
Allow_Default_Join	vrai (1)	Tableau 368
Target_NWK_ID	0	Tableau 368
Target_NWK_BitMask	0xFFFF	Tableau 368
Target_Join_Method	1 (clé asymétrique)	Tableau 368
Nombre de PKI_Certificates	1	Tableau 368
Current_UTC_Adjustment	35	Tableau 368
White_List	[ ]	Tableau 371
Symmetric_Key_List	{K_global}	Tableau 371
Symmetric_Key_Expiry_Times	{0xFFFF FFFF}	Tableau 371
Target_NWK_ID	0	Tableau 371
Target_Join_Method	1 (clé asymétrique)	Tableau 371
Target_Join_Time	0	Tableau 371
Allow_Provisioning	vrai (1)	Tableau 371
Allow_Default_Join	vrai (1)	Tableau 371
Device_Specific_Provisioning_Flag	désactivé (0)	Tableau 371
DPSO_Alerts_AlertDescriptor	[faux, 6]	Tableau 371
Current_UTC_Adjustment	35	Tableau 371
Device_EUI	0x0000 0000 0000 0001	Tableau 372
Device_Symmetric_Key	K_global	Tableau 372
Device_Symmetric_Key_Expiry_Time	{0xFFFF FFFF}	Tableau 372
Target_NWK_ID	0	Tableau 372
Target_Join_Method	1 (clé asymétrique)	Tableau 372
Allow_Provisioning	vrai	Tableau 372
Allow_Default_Join	vrai	Tableau 372

D.9 Passerelle (informative)

Le Tableau D.8 énumère les valeurs par défaut de la configuration de la passerelle.

Tableau D.8 – Valeurs par défaut de la configuration de la passerelle

Nom	Valeur par défaut initiale	Référence
Max_Devices	0	Tableau U.41

Annexe E
(informative)

Utilisation de réseaux dorsaux

E.1 Généralités

Le réseau dorsal peut être d'une utilisation avantageuse pour le concepteur de système, car il enlève le message du moyen de terrain de Type A, octroyant de la largeur de bande supplémentaire et une QoS plus élevée pour d'autres messages.

E.2 Caractéristiques recommandées

Bien que la dorsale elle-même ne soit pas spécifiée dans la présente norme, il est admis et recommandé que la dorsale présente les caractéristiques suivantes:

- Débit supérieur ou égal au débit du moyen de terrain de Type A (≥ 250 kbit/s).
- Capacité de prendre en charge le trafic bidirectionnel de message non sollicité.
- Qualité de service d'un niveau suffisant permettant de maintenir la synchronisation du temps de réseau. Cela peut mettre en place des méthodes spécifiques de synchronisation du temps sur la dorsale.
- Fiabilité élevée. Il convient que le fonctionnement ne charge pas le réseau avec des fréquents messages perdus et de fréquentes répétitions de tentative.
- Sécurité suffisante pour ne pas présenter une menace de sécurité pour l'application des utilisateurs finaux ou pour le réseau.
- Capacité d'encapsuler les TPDU ou TSDU de protocoles (de tunnellisation) définies par la présente norme ou de les convertir afin qu'elles puissent traverser la dorsale sans être modifiées lorsqu'elles émergent aux appareils dorsaux. Généralement, la dorsale doit être capable de prendre une TSDU conforme à la norme au point d'entrée et la livrer, inchangée, à travers la dorsale vers le point de sortie.
- Capacité de préserver les mécanismes de sécurité d'application de bout en bout.
- Prise en charge du réseautage multipoint entre les appareils.

Il est admis que de nombreux bus de terrain normalisés peuvent ne pas avoir ces caractéristiques et peuvent donc ne pas être adaptés à être utilisés comme réseau dorsal. Dans de nombreux cas, un réseau dorsal sera un réseau IP tel que selon l'ISO/IEC 8802-3 (IEEE 802.3) ou l'ISO/IEC 8802-11 (IEEE 802.11), mais il n'y a aucune exigence pour cela. Il existe beaucoup d'autres variantes sur le marché qui sont bien adaptées pour les besoins d'un réseau dorsal. Ceux-ci pourraient inclure des réseaux sans fil point à point ou bien point à multipoint.

E.3 Dorsales de protocole internet

E.3.1 Méthodes d'émission d'unités de données de protocole IPv6

Dans de nombreux cas, une dorsale disponible utilisera une NL de protocole internet (IP). Dans ce cas, il existe plusieurs manières différentes de transporter les TPDU du réseau de capteurs industriels sans fil (WISN) en utilisant le comportement de protocole normalisé:

- Encapsuler au sein de NPDU IPv4 les unités de données de protocole de transport du réseau de capteurs industriels sans fil.

  Le mécanisme utilisé pour encapsuler les TPDU du WISN au sein de NPDU IPv4 est formellement connu comme étant IPv6 sur IPv4 ou 6over4 et s'appelle parfois

Ethernet virtuel. Cette méthode est documentée dans l'ETF RFC 2529. Un routeur dorsal – l'IETF RFC 2529 les appelle hôtes IPv6 – situé sur une liaison physique qui n'a aucun routeur IPv6 directement connecté peut devenir un hôte IPv6 complètement fonctionnel en utilisant un domaine de multidiffusion IPv4 comme sa liaison locale virtuelle. Les routeurs dorsaux connectés en utilisant cette méthode n'exigent pas d'adresses compatibles IPv4 ou de tunnels configurés.

- Tunnelliser sur le réseau IPv4 les unités de données de protocole de transport du réseau de capteurs industriels sans fil.

  En suivant l'IETF RFC 4213, cette méthode encapsule des unités de données de protocole (PDU) IPv6 au sein d'en-têtes IPv4 pour les transporter sur des infrastructures de routage IPv4. Deux types de tunnellisation sont possibles, à savoir le type configuré et le type automatique. Dans la tunnellisation configurée, l'adresse de point d'extrémité de tunnel IPv4 est déterminée par les informations de configuration relatives au nœud d'encapsulation. Dans la tunnellisation automatique, l'adresse de point d'extrémité du tunnel IPv4 est déterminée à partir de l'adresse IPv4 intégrée dans l'adresse de destination compatible IPv4 de la PDU IPv6.

- Encapsuler au sein de NPDU Ethernet brutes les unités de données de protocole de transport du réseau de capteurs industriels sans fil.

  Cette méthode spécifie le format NPDU pour la transmission des PDU IPv6 à la suite de l'IETF RFC 2464. En outre, cette méthode dicte la formation d'adresses IPv6 locales à une liaison et d'adresses configurées sans états sur les réseaux Ethernet. Finalement, cette approche spécifie le contenu des adresses de couche liaison source/cible utilisées dans la sollicitation de routeur, l'annonce de routeur, la sollicitation de voisin, l'annonce de voisin, et redirige les messages lorsque ces messages sont émis sur un réseau Ethernet.

- Utiliser la dorsale IPv6 native sans la moindre encapsulation ou tunnellisation.

  Si la dorsale utilise une NL IPv6, ni l'encapsulation ni la tunnellisation ne sont nécessaires, car le mode natif d'une dorsale est de transporter des PDU IPv6.

E.3.2 Découverte d'appareils homologues d'un routeur dorsal

Pour que le routeur dorsal (BBR) fonctionne correctement et connecte des appareils de WISN sur la dorsale, il a besoin de connaître les adresses dorsales des autres BBR ou homologues dans le réseau dorsal. Au sein de chaque BBR, il convient que les informations d'adressage de ses pairs soient stockées dans une table d'homologues de routeur dorsal (BRPT). Il existe deux méthodes de base de génération de la BRPT, à savoir la configuration et la découverte.

NOTE La BRPT et le mécanisme pour découvrir des homologues ne relèvent pas du domaine d'application de la présente norme.

La configuration se produit lorsque les adresses des BBR homologues sont insérées dans la BRPT par le gestionnaire de système ou par un opérateur. Les avantages de cette méthode sont qu'elle est directe et empêche le BBR d'accéder à des appareils inappropriés sur la dorsale.

La découverte se produit lorsque le BBR recherche automatiquement des appareils homologues dans la dorsale. Il existe plusieurs techniques de découverte, telles que celles mentionnées dans l'IETF RFC 2529 et autres. Les avantages de cette méthode sont qu'elle est automatique, n'exige aucune implication d'opérateur, et peut être mise à jour facilement et souvent.

E.3.3 Sécurité

E.3.3.1 Sécurité des unités de données de protocole de transport

Les mécanismes de sécurité de la dorsale ne relèvent pas du domaine d'application de la présente norme. Les méthodes de sécurité IP typiques comprennent IPSec, SSL, et d'autres. En plus de tous les éventuels mécanismes de sécurité sur la dorsale, le mécanisme de sécurité de la TL du WISN protège la TPDU au sein de la dorsale.

E.3.3.2 Sécurité de la dorsale

Selon la perception de certains, le fait d'autoriser un WISN à accéder à une dorsale d'IP pourrait dégrader la sécurité de la dorsale. Ce souci peut être atténué en limitant l'accès de BBR aux seuls BBR homologues par l'intermédiaire d'une liste de contrôle d'accès ou par l'utilisation de pare-feu configurés pour limiter correctement l'accès à des appareils spécifiques.

Annexe F
(normative)

Concepts de sécurité de base – Notation et représentation

F.1 Chaînes et opérations sur les chaînes

Une chaîne est une séquence de symboles sur un ensemble spécifique (par exemple, l'alphabet binaire (0, 1) ou l'ensemble de tous les octets).

La taille d'une chaîne est le nombre de symboles qu'elle contient (sur le même alphabet).

La concaténation à droite de deux chaînes x et y de taille m et n respectivement (notation $x \,||\, y$) est la chaîne z de taille $m + n$ qui coïncide avec x sur ses m symboles de gauche et avec y sur ses n symboles de droite.

Un octet est une chaîne de symboles de longueur 8. Dans ce contexte, tous les octets sont des chaînes sur l'alphabet binaire.

F.2 Entiers, octets et leur représentation

Tout au long de la présente norme, la représentation des nombres entiers sous la forme de chaînes d'octets et celle des chaînes d'octets sous la forme de chaînes binaires doivent être fixe.

Tous les nombres entiers doivent être représentés sous la forme de chaînes d'octets dans l'ordre "octet de poids fort en premier". Cette représentation est conforme à la convention donnée dans l'ANSI X9.63:2011, 4.3.

Tous les octets doivent être représentés sous la forme de chaînes binaires dans l'ordre "bit de poids fort en premier".

F.3 Entités

Tout au long de la présente norme, chaque entité doit être un DEV et doit être identifié de façon univoque par son adresse EUI64. Le paramètre entityIdSize doit avoir la valeur 64.

Annexe G
(informative)

Utilisation de chaînes de certificats pour la configuration par liaison radio

La présente norme utilise le certificat implicite appelé le PublicReconstrKey (voir Annexe H pour les détails) pour la cryptographie à clés asymétriques. Connaissant l'identité d'un appareil A (ID_A) et le certificat implicite γ_A de l'appareil, la clé publique de l'appareil A peut être calculée en utilisant l'équation suivante:

$$Q_A = Hash(\gamma A||ID_A)\gamma A+Q_{CA}$$

où Q_{CA} est la clé publique de l'autorité de certification (CA).

Dans ce contexte, les étapes suivantes décrivent le processus pour la configuration par OTA en utilisant la cryptographie à clés 'asymétriques telle que présentée à la Figure 137.

1) Le CA édite Q_{CA}, sa clé publique, sur le web.
2) Le fabricant d'appareil (device manufacturer (DM)) obtient un certificat issu de la CA:

   C_DM = PublicReconstrKey(DM) || Subject(DM) || Issuer(CA) || Text

   où:

   - Subject = ID du DM
   - Issuer = ID de la CA
   - PublicReconstrKey_DM = γ_DM est utilisé pour calculer la clé publique du DM en utilisant l'équation:

     $Q_{DM} = HASH(\gamma\_DM||Subject)\gamma\_DM + Q_{CA}$
3) L'appareil individuel obtient un certificat issu du DM:

   C_DEV = PublicReconstrKey(DEV) || Subject(DEV) || Issuer(DM) || Text

   où:

   - Subject = ID de l'appareil
   - Issuer =ID du DM
   - PublicReconstrKey_DEV = γ_DEV est utilisé pour calculer la clé publique de l'appareil en utilisant l'équation:

     $Q_{DEV} = HASH(\gamma\_DEV||Subject)\gamma\_DEV + Q_{DM}$
   - C_DEV et C_DM sont remplis dans le DBP par le DM.
4) Le DBP rattache le PD dans un réseau de configuration. Le PD a le QCA.
5) Le DBP envoie un numéro aléatoire, C_DEV et C_DM au PD. Le PD calcule Q_DEV comme expliqué aux étapes 2) et 3).
6) Un mécanisme de défi/réponse est utilisé pour authentifier l'appareil, et il convient que le gestionnaire de sécurité valide le certificat implicite du fabricant à ce stade.
7) Si le défi/réponse donne un résultat positif, le PD envoie K_join chiffré avec Q_DEV.

Annexe H
(normative)

Blocs modules de base de sécurité

H.1 Blocs modules de base cryptographiques à clés symétriques

H.1.1 Vue d'ensemble

Les primitives et les éléments de données cryptographiques à clés symétriques suivants sont définis pour une utilisation avec toutes les opérations de traitement de la sécurité spécifiées dans la présente norme.

H.1.2 Paramètres de domaine des clés symétriques

La clé symétrique doit avoir la taille de clé keylen=128 (en bits).

H.1.3 Cryptage par blocs

Le chiffrement de bloc utilisé dans la présente norme doit être AES-128, comme spécifié dans l'ISO/IEC 18033-3. Ce chiffrement de bloc doit être utilisé avec des clés symétriques comme spécifié en H.1.2. Dans ce cas, la taille de clé est égale à la taille de bloc du bloc chiffrant, 128 bits.

H.1.4 Mode de fonctionnement

Le mode de fonctionnement de cryptage par blocs utilisé dans la présente norme doit être le mode de fonctionnement CCM*, tel que spécifié dans l'IEEE 802.15.4:2011, B.3.2.

H.1.5 Fonction de hachage cryptographique

La fonction de hachage cryptographique utilisée dans la présente norme doit être la fonction de hachage cryptographique à cryptage par blocs spécifiée à l'Article H.9, avec les instanciations suivantes:

- chaque entité doit utiliser le cryptage par blocs tel que spécifié en H.1.3;
- tous les nombres entiers et tous les octets doivent être représentés tels que spécifiés à l'Article F.2.

La fonction de hachage de Matyas-Meyer-Oseas (voir Article H.9) a une taille de compilation de messages hashlen qui est égale à la taille de bloc, en bits, du bloc chiffrant établi.

H.1.6 Fonction de hachage codée pour authentification de message

Le code d'authentification de message par hachage (HMAC) codé utilisé dans la présente norme doit être HMAC, tel que spécifié dans la FIPS 198, avec les instanciations suivantes:

- chaque entité doit utiliser la fonction de hachage cryptographique H telle que spécifiée en H.1.5;
- la taille de bloc B doit avoir la valeur entière B=keylen/8, où keylen est tel que spécifié en H.1.2 (à savoir, B est égal à la taille de la clé symétrique, en octets, qui est utilisée par la fonction de hachage codée);
- la taille de sortie HMAClen de la fonction de HMAC doit avoir la même valeur entière que le paramètre de compilation de messages hashlen, tel que spécifié en H.1.5.

H.1.7 Fonction de hachage codée spécialisée pour authentification de message

Le code d'authentification de message par hachage codé spécialisé[14] utilisé dans la présente norme doit être le code d'authentification de message par hachage codé, tel que spécifié en H.1.6.

H.1.8 Paramètres de domaine de défi

Les paramètres de domaine de défi utilisés dans la présente norme doivent être tels que spécifiés en H.6.2, avec l'instanciation (minchallengelen, maxchallengelen)=(keylen, keylen), où keylen est tel que spécifié en H.1.2.

Tous les défis doivent être validés en utilisant la primitive de validation de défi telle que spécifiée à l'Article H.7.

H.2 Blocs modules de base cryptographiques à clés asymétriques

H.2.1 Généralités

Les primitives et les éléments de données cryptographiques à clés asymétriques suivants sont définis pour une utilisation avec toutes les opérations de traitement de la sécurité spécifiées dans la présente norme.

NOTE Voir également l'ISO/IEC 180332 pour plus d'informations relatives à la cryptographie asymétrique.

H.2.2 Paramètres de domaine de courbe elliptique

Les paramètres de domaine de courbe elliptique utilisés dans la présente spécification doivent être ceux pour la courbe ansit283k1 telle que spécifiée dans la norme ANSI X9.63:2011, appendice J4.5, exemple 1.

Tous les points de courbe elliptique doivent être validés en utilisant la primitive de validation de clé publique telle que spécifiée dans l'ANSI X9.63:2011, 5.2.2.

H.2.3 Représentation de points de courbe elliptique

Tous les points de courbe elliptique doivent être représentés dans la notation polynomiale telle spécifiée dans l'ANSI X9.63:2011, 4.1.2.1. Tous les points de courbe elliptique doivent être émis sous forme compressée, telle que spécifiée dans l'ANSI X9.63:2011, 4.2.2.

H.2.4 Paire de clés publiques d'une courbe elliptique

Une paire de clés de courbe elliptique est constituée d'un nombre entier q et d'un point Q sur la courbe déterminée en multipliant le point générateur G de la courbe par ce nombre entier (à savoir Q=qG) comme spécifié dans la norme ANSI X9.63:2011. Ici, Q est appelé la clé publique, alors que q est appelé la clé privée; la paire (q, Q) est appelée la paire de clés publiques. Chaque clé privée doit être représentée comme spécifié dans l'ANSI X9.63:2011, 4.3.1. Chaque clé privée doit être sur la courbe comme spécifié en H.2.2 et doit être représentée comme spécifié en H.2.3.

---

14 Cela se réfère au schéma MAC où la fonction MAC a la propriété complémentaire d'être également une pré-image et résistante aux collisions pour les parties prenantes connaissant la clé (voir également la remarque 9.8 de Menezes et al.). De telles fonctions MAC permettent la dérivation de clé dans des contextes où une commande unilatérale des clés est indésirable.

H.3 Information de codage

H.3.1 Généralités

Ce qui suit spécifie le format des informations de codage à clés asymétriques utilisées dans la présente norme.

H.3.2 Certificats implicites de cryptographie sur courbe elliptique

Les certificats implicites IC_U doivent être spécifiés comme

IC_U = PublicKeyReconstrData || Subject || Issuer || Usage_Serial || KeyValidityInfo || Text

où:

- Le paramètre PublicKeyReconstrData doit représenter les données de reconstruction de clés publiques BEU telles que spécifiées dans le plan de génération de certificats implicites (voir H.5.1).
- Le paramètre Subject doit être l'entité U qui est liée aux données de reconstruction de clés publiques BEU pendant l'exécution du plan de génération de certificats implicites, à savoir l'entité qui est prétendument propriétaire de la clé privée correspondant à la clé publique qui peut être reconstruite à partir de PublicReconstrKey.
- Le paramètre Issuer doit être l'entité de l'autorité de certification (CA) qui crée le certificat implicite pendant l'exécution du plan de génération de certificats implicites.
- Le paramètre Usage_Serial est défini dans le Tableau 68.
- Le paramètre KeyValidityInfo doit indiquer la période de validité du support de codage telle qu'indiquée par les paramètres ValidNotBefore et ValidNotAfter, qui indiquent respectivement le début et la fin de la période de validité. Le KeyValidityInfo doit être formaté comme

  KeyValidityInfo = ValidNotBefore || ValidNotAfter

  où ValidNotBefore et ValidNotAfter doivent être représentés tels que spécifiés en 12.22.4.2.
- Le paramètre Text doit être la représentation d'informations complémentaires, telles que spécifiées en H.3.4.
- La chaîne I_U telle que spécifiée dans le plan de génération de certificats implicites (voir H.5.1) doit être la chaîne d'octets comprenant les chaînes Subject, Issuer, et Text, comme suit:

  I_U = Subject || Issuer || Text

H.3.3 Certificats manuels de cryptographie sur courbe elliptique

Les certificats manuels MC_U doivent être spécifiés sous la forme MC_U = PublicKey || Subject || Issuer || Text, où:

- Le paramètre PublicKey doit être la représentation en octet de la clé publique W_U telle que spécifiée dans la transformation de la génération de certificats manuels.
- Le paramètre Subject doit être l'entité U du prétendu propriétaire de la clé privée correspondant à la clé publique représentée par PublicKey.
- Le paramètre Issuer doit être l'entité de la CA qui crée le certificat manuel pendant l'exécution de la transformation de génération de certificats manuels (ladite autorité de certification).
- Le paramètre Usage_Serial est défini dans le Tableau 68.
- Le paramètre KeyValidityInfo doit indiquer la période de validité du support de codage telle qu'indiquée par les paramètres ValidNotBefore et ValidNotAfter, qui indiquent

respectivement le début et la fin de la période de validité. Le KeyValidityInfo doit être formaté comme

KeyValidityInfo = ValidNotBefore || ValidNotAfter

où ValidNotBefore et ValidNotAfter doivent être représentés tels que spécifiés en 12.22.4.2.

- Le paramètre Text doit être la représentation d'informations complémentaires, telles que spécifiées en H.3.4.
- La chaîne I_U telle que spécifiée dans le plan de génération de certificats manuels (voir l'Article H.10) doit être la chaîne d'octets comprenant les chaînes Subject, Issuer, et Text, comme suit:

  I_U = Subject || Issuer || Text

NOTE Un certificat manuel n'est pas un certificat numérique réel, car la liaison entre les paramètres PublicKey et Subject est établie et vérifiée par des moyens non cryptographiques.

H.3.4 Informations supplémentaires

Les informations complémentaires Text doivent être spécifiées comme suit:

Text = Reserved

où le paramètre Reserved permet la prise en charge de futures extensions des informations complémentaires et doit être mis à une chaîne de bits zéro partout pour la présente version de la norme.

H.4 Plan d'agrément de clés

H.4.1 Plan d'agrément de clé à clés symétriques

Le plan d'agrément de clés à clés symétriques utilisé dans la présente norme doit être le plan complet de confirmation de clé à clés symétriques tel que spécifié avec les instanciations suivantes:

- Chaque entité doit être identifiée telle que spécifiée à l'Article F.3.
- Chaque entité doit utiliser le plan HMAC tel que spécifié en H.1.5.
- Chaque entité doit utiliser la fonction de hachage cryptographique telle que spécifiée en H.1.5.
- Le paramètre keydatalen doit avoir la même valeur entière que le paramètre de taille de clé keylen tel que spécifié en H.1.2.
- Chaque entité doit utiliser les paramètres de domaine de défi tels que spécifiés en H.1.8.
- Tous les octets doivent être représentés tels que spécifiés à l'Article F.2.

H.4.2 Plan d'agrément de clé à clés asymétriques

Le plan d'agrément de clés à clés asymétriques utilisé dans la présente norme doit être le MQV complet avec plan de confirmation de clé tel que spécifié dans l'ANSI X9.63:2011, 6.11, avec les instanciations suivantes:

- Chaque entité doit être identifiée telle que spécifiée à l'Article F.3.
- Chaque entité doit utiliser le plan HMAC tel que spécifié en H.1.5.
- Chaque entité doit utiliser la fonction de hachage cryptographique telle que spécifiée en H.1.5.
- Le paramètre keydatalen doit avoir la même valeur entière que le paramètre de taille de clé keylen tel que spécifié en H.1.2.

- Le paramètre SharedData doit être la chaîne vide. Le paramètre shareddatalen doit avoir la valeur entière 0.
- Chaque entité doit utiliser les paramètres de domaine de courbe elliptique tels que spécifiés en H.2.2.
- Tous les points de courbe elliptique doivent être représentés tels que spécifiés à l'article H.2.3.
- Tous les octets doivent être représentés tels que spécifiés à l'Article F.2.

H.5 Plans d'informations de codage

H.5.1 Plan de certificats implicites

Le plan de certificats implicites utilisé dans la présente norme doit être le plan de certificats implicites ECQV tel que spécifié en SEC 4, avec les instanciations suivantes:

- Chaque entité doit être identifiée telle que spécifiée à l'Article F.3.
- Chaque entité doit utiliser la fonction de hachage cryptographique telle que spécifiée en H.1.5.
- Chaque entité doit utiliser les paramètres de domaine de courbe elliptique tels que spécifiés en H.2.2.
- Tous les points de courbe elliptique doivent être représentés tels que spécifiés à l'article H.2.3.
- Tous certificats implicites doivent être représentés tels que spécifiés en H.3.2.
- L'infrastructure des certificats implicites doit être l'un des plans spécifiés en H.3.2.
- Tous les octets doivent être représentés tels que spécifiés à l'Article F.2.

H.5.2 Plan de certificats manuels

Le plan de certificats manuels utilisé dans la présente norme doit être le plan de certificats manuels tel que spécifié à l'Article H.10, avec les instanciations suivantes:

- Chaque entité doit être identifiée telle que spécifiée à l'Article F.3.
- Chaque entité doit utiliser les paramètres de domaine de courbe elliptique tels que spécifiés en H.2.2.
- Tous les points de courbe elliptique doivent être représentés tels que spécifiés à l'article H.2.3.
- Tous certificats manuels doivent être représentés tels que spécifiés en H.3.2.
- L'infrastructure des certificats manuels doit être l'un des plans spécifiés en H.3.2.
- Tous les octets doivent être représentés tels que spécifiés à l'Article F.2.

H.6 Génération et validation des paramètres de domaine de défi

H.6.1 Vue d'ensemble

Les paramètres de domaine de défi imposent des contraintes sur la(les) taille(s) des défis de bit qu'un plan attend. A ce titre, cela détermine une limite sur l'entropie des défis et, donc, sur la sécurité des plans cryptographiques dans lesquels ces défis sont utilisés. Dans la plupart des plans, les paramètres de domaine de défi seront tels que seuls les défis d'une taille fixe seront acceptés (par exemple, défis de 128 bits). Toutefois, on peut définir les paramètres de domaine de défi tels que des défis de taille variable soient acceptés. Ce dernier cas est utile dans des contextes où les entités qui souhaitent s'engager dans des plans cryptographiques pourraient avoir un générateur de bits aléatoire défectueux ou de basse qualité. Le fait de permettre aux deux entités qui s'engagent dans un plan d'apporter comme contribution de longues entrées permet à chacune d'apporter de l'entropie suffisante au plan à proximité.

Dans la présente norme, des paramètres de domaine de défi seront partagés par un certain nombre d'entités utilisant un plan de la norme. Les paramètres de domaine de défi peuvent être publics; la sécurité du système ne repose pas sur le fait que ces paramètres soient secrets.

H.6.2 Génération de paramètres de domaine de défi

Les paramètres de domaine de défi doivent être générés en utilisant la routine suivante:

- Entrée: Cette routine ne prend aucune entrée.
- Actions: Les actions suivantes sont entreprises:
  - Choisir deux nombres entiers non négatifs minchallengelen et maxchallengelen, tels que minchallengelen ≤ maxchallengelen.
- Sortie: Paramètres de domaine de défi D=(minchallengelen, maxchallengelen).

H.6.3 Vérification de paramètres de domaine de défi

Les paramètres de domaine de défi doivent être vérifiés en utilisant la routine suivante:

- Entrée: Ensemble prétendu de paramètres de domaine de défi D = (minchallengelen, maxchallengelen).
- Actions: Les vérifications suivantes sont faites:
  - La vérification que minchallengelen et maxchallengelen sont des nombres entiers non négatifs.
  - Vérifier que minchallengelen ≤ maxchallengelen.
- Sortie: Si l'une des vérifications ci-dessus a échoué, alors produire invalide et rejeter les paramètres de domaine de défi. Autrement, produire valide et accepter les paramètres de domaine de défi.

H.7 Primitive de validation de défi

La validation de défi se réfère au processus de vérification des propriétés de taille d'un défi. Elle est utilisée pour vérifier si, oui ou non, un défi devant être utilisé par un plan dans la norme a la taille suffisante (par exemple, des messages qui sont trop courts sont rejetés, en raison d'entropie insuffisante).

La primitive de validation de défi est utilisée à l'Article H.7 et utilise la routine suivante:

- Entrée: L'entrée de la transformation de validation est un ensemble valide de paramètres de domaine de défi D = (minchallengelen, maxchallengelen), ainsi que la chaîne binaire Challenge.
- Actions: Les actions suivantes sont entreprises:
  - Calculer la longueur de bits challengelen de la chaîne de bits Challenge.
  - Vérifier que challengelen ∈ [minchallengelen, maxchallengelen]. (A savoir, vérifier que le défi a une taille appropriée.)
- Sortie: Si la vérification ci-dessus échoue, alors produire invalide et rejeter le défi. Autrement, produire valide et accepter le défi.

H.8 Primitive de génération de clés secrètes (SKG)

La primitive SKG dérive une valeur secrète partagée à partir d'un défi possédé par une entité U_1 et d'un défi possédé par une entité U_2 si tous les défis partagent les mêmes paramètres de domaine de défi. Si les deux entités exécutent correctement cette primitive avec des défis correspondants comme entrées, la même valeur secrète partagée sera produite.

La valeur secrète partagée doit être calculée comme suit:

- Conditions préalables: Ce qui suit représente les conditions préalables pour l'utilisation de la primitive SKG:
  - Chaque entité doit être liée à un identificateur unique (par exemple, noms distingués). Tous les identificateurs doivent être des chaînes de bits de la même taille, entityIdSize. L'identificateur de l'entité U_1 sera désigné par la chaîne de bits U_1. L'identificateur de l'entité U_2 sera désigné par la chaîne de bits U_2.
  - Un plan MAC spécialisé[15] doit avoir été sélectionné, avec les informations d'étiquetage telles que spécifiées dans l'ANSI X9.63:2011, 5.7.1. La taille en bits des clés utilisées par le plan MAC spécialisé est désignée par macKeySize.
- Entrée: La primitive SKG prend comme entrée:
  - Une chaîne de bits MacKey de taille macKeySize bits devant être utilisée comme la clé d'un plan MAC spécialisé établi.
  - Une chaîne de bits QEU_1 appartenant au propriétaire U_1.
  - Une chaîne de bits QEU_2 appartenant au propriétaire U_2.
- Actions: Les actions suivantes sont entreprises:
  - Former la chaîne de bits constituée de l'identificateur de U_1, de l'identificateur de U_2, la chaîne de bits QEU_1 correspondant au défi de U_1, et la chaîne de bits QEU_2 correspondant au défi de U_2:
  - MacData = $U_1 \,||\, U_2 \,||\, QEU_1 \,||\, QEU_2$.
  - Calculer l'étiquette MacTag pour MacData avec la clé MacKey en utilisant la transformation d'étiquetage du plan MAC spécialisé établi:
  - $MacTag = MAC_{MacKey}(MacData)$.
  - Si la transformation d'étiquetage produit non valide, produire également non valide et arrêter.
  - Mettre Z=MacTag.
- Sortie: La chaîne de bits Z comme la valeur secrète partagée.

H.9 Fonction de hachage cryptographique à chiffrement par blocs

La fonction de hachage de Matyas-Meyer-Oseas est une fonction de hachage cryptographique basée sur des blocs chiffrants. Cette fonction de hachage est définie pour des blocs chiffrants avec une taille de clé qui est égale à la longueur du bloc, comme l'AES-128, et avec un choix particulier pour le vecteur d'initialisation fixe IV (qui est défini ici comme étant IV=0).

NOTE Pour une définition plus générale de la fonction de hachage de Matyas-Meyer-Oseas, voir le *Handbook of applied cryptography*:1996, 9.4.1, inclus dans la Bibliographie.

La fonction de hachage est définie comme suit:

- Conditions préalables: Ce qui suit représente les conditions préalables au fonctionnement de la fonction de hachage de Matyas-Meyer-Oseas:
  - Une fonction E de chiffrement par blocs doit avoir été choisie, avec une taille de clé qui est égale à la taille de bloc. La fonction de hachage de Matyas-Meyer-Oseas a une taille de compilation de messages qui est égale à la taille de bloc de la fonction de chiffrement établie. Elle opère sur des chaînes de bits de taille inférieure à 2^n, où n est la taille de bloc, en octets, du chiffrement par blocs établi.

---

15 Cela se réfère au schéma MAC dans lequel la fonction MAC a la propriété complémentaire d'être également une pré-image et résistante aux collisions pour les parties prenantes connaissant la clé (voir également la remarque 9.8 de Menezes et al.). De telles fonctions MAC permettent la dérivation de clé dans des contextes où une commande unilatérale des clés est indésirable.

  - Une représentation fixe de nombres entiers sous la forme de chaînes de bits ou de chaînes d'octets doit avoir été choisie.

- Entrée: L'entrée à la fonction de hachage de Matyas-Meyer-Oseas est comme suit:
  - Une chaîne de bits M de taille de bits l bits, où $0 \leq l < 2^n$.
- Actions: La valeur de hachage doit être dérivée comme suit:
  - Bourrer le message M selon la méthode suivante:
  - Concaténer à droite au message M la valeur binaire constituée d'un bit de 1 suivi de k bits de 0, où k est la plus petite solution non négative à l'équation

    $l + 1 + k \equiv 7n \pmod{8n}$.
  - Former de le message bourré M en concaténant à droite à la chaîne résultante la chaîne de n bits qui est égale à la représentation binaire du nombre entier l.
  - Analyser le message bourré M comme étant $M_1 \| M_2 \| \ldots \| M_t$ où chaque bloc de message M_i est une chaîne de n octets.
  - La sortie $Hash_t$ est définie par

    $Hash_0 = 0^{8n}$; $Hash_j = E(Hash_{j-1}, M_j) \oplus M_j$ pour $j=1,\ldots,t$.

Ici, E (K, x) est le cryptogramme qui résulte du chiffrement du texte en clair x, en utilisant la fonction E de chiffrement à blocs chiffrants établie avec la clé K; la chaîne 0^{8n} est la chaîne de bits 0 partout de n octets.

- Sortie: La chaîne de bits Hasht comme la valeur de hachage.

La fonction cryptographique de hachage opère sur une force binaire de longueur inférieure à 2^n bits, où n est la taille de bloc (ou taille de clé) du chiffrement par blocs établi, en octets. Par exemple, la fonction de hachage de Matyas-Meyer-Oseas avec AES-128 opère sur les chaînes binaires de taille inférieure à 2^{16} bits. Il est supposé que tous les appels de fonction de hachage sont sur les chaînes binaires de taille inférieure à 2^n bits. Tout plan tentant d'appeler la fonction de hachage sur une chaîne de bits dépassant 2^n bits doit produire invalide et s'arrêter.

H.10 Plan de certificats manuels à cryptographie sur courbes elliptiques

H.10.1 Vue d'ensemble

Un plan de certificat manuel basé sur la cryptographie sur courbes elliptiques (ECC) qui est utilisée dans la présente norme est décrit.

Le plan de certificat manuel est utilisé par trois entités: une autorité de certification CA, un demandeur de certificat U, et un processeur de certificat V, où U souhaite obtenir un certificat manuel issu de CA afin d'acheminer vers V la clé publique associée de U.

Le plan de certificat manuel est décrit en termes d'une transformation de génération de certificat et d'une transformation de traitement de certificat. CA, U, et V utilisent ces plans lorsqu'ils souhaitent communiquer.

Avant l'utilisation du plan, U, V, et CA s'accordent sur les paramètres avec lesquels le plan doit être utilisé. En particulier, cela inclut que U et V obtiennent une copie authentique de l'identificateur unique de CA.

CA exécute la transformation de génération de certificat manuel pour calculer une paire de clés publiques sur courbe elliptique pour U et un certificat manuel MC pour cette clé publique fournie par CA. V exécute la transformation de traitement du certificat manuel, pour obtenir la clé publique statique prétendue de U à partir du certificat manuel prétendu MC de U présenté à V.

La transformation de génération de certificat manuel produit une paire de clés publiques et un certificat pour cette clé publique. Cette paire de clés publiques doit être communiquée au prétendu détenteur d'une manière sécurisée et authentique. Le mécanisme par lequel cette paire de clés publiques est communiquée ne relève pas du domaine d'application de la présente norme.

La transformation de traitement de certificat manuel produit une clé publique statique (et les informations de codage associées) destinée au détenteur revendiqué; la preuve que cette clé publique est véritablement liée à cette entité ne peut cependant pas être corroborée par l'intermédiaire du traitement du certificat manuel. Ainsi, avec les certificats manuels, la liaison d'une entité et de sa clé publique ou privée ne peut pas être vérifiée, bien qu'on puisse obtenir la preuve qu'une certaine entité qui prétend être liée à la clé publique a en effet accès à la clé privée correspondante, pendant l'utilisation cryptographique de la clé publique (par exemple, par l'intermédiaire de l'exécution d'un plan d'agrément de clé authentifié ou d'une transformation de signature impliquant cette paire de clés publiques).

La transformation de génération de certificat manuel est spécifiée en H.10.2 et la transformation de traitement de certificat manuel est spécifiée en H.10.3.

Les conditions préalables pour l'utilisation du plan sont:

- Une infrastructure doit avoir été établie pour l'opération du plan, y compris un format de certificat, des règles de traitement de certificats, et des identificateurs uniques. Pour un exemple d'une telle infrastructure, voir l'IETF RFC 3280.
- Chaque entité a une copie authentique des paramètres de domaine de courbe elliptique de systèmes D=(p,a,b,G,n,h) ou D=(m,f(x),a,b,G,n,h). Ces paramètres doivent avoir été générés en utilisant les primitives de génération de paramètre indiquées en SEC 1:2009, 3.1.1.1 ou 3.1.2.1. En outre, les paramètres doivent avoir été validés en utilisant les primitives de validation de paramètre en SEC 1:2009 3.1.1.2 ou 3.1.2.2.
- Chaque entité doit être liée à un identificateur unique (par exemple, noms distingués). Tous les identificateurs doivent être des chaînes de bits de la même taille, entityIdSize. L'identificateur de l'entité U sera désigné par la chaîne de bits U. L'identificateur de l'entité V sera désigné par la chaîne de bits V. L'identificateur de l'entité CA sera désigné par la chaîne de bits CA.
- Une fonction de hachage cryptographique Hash doit avoir été choisie pour une utilisation avec le plan de génération de certificat implicite ECQV. Soit hashlen désignant la taille en bits de la valeur de sortie de cette fonction de hachage.
- Chaque entité doit avoir décidé de la manière de représenter les points de courbe elliptique sous forme de chaînes d'octets (à savoir forme compressée, forme non compressée, ou forme hybride).
- Une représentation fixe des octets sous la forme de chaînes binaires doit avoir été choisie (par exemple, ordre "bit de poids fort en premier" ou ordre "bit de poids faible en premier").

H.10.2 Transformation de génération de certificat manuel à cryptographie sur courbes elliptiques

Une CA doit exécuter la transformation suivante pour fournir un certificat manuel pour l'utilisateur, U. La CA doit obtenir une copie authentique de l'identificateur de U.

- Entrées: Cette routine ne prend aucune entrée.
- Ingrédients: La transformation de génération de certificat utilise la primitive de génération de paire de clés donnée en SEC 1:2009, 3.2.1, et la primitive de génération de certificat manuel de l'infrastructure établie.
- Actions: La CA doit poursuivre comme suit:

  - La primitive de génération de paire de clés spécifié en SEC 1:2009, 3.2.1, doit être utilisée pour générer une paire de clés éphémères (w_U, W_U) pour les paramètres D.
  - Le point de courbe elliptique W_U doit être converti en la chaîne d'octets WE_U telle que spécifiée en SEC 1:2009, 2.3.3.
  - La chaîne d'octets I_U, qui représente les données du certificat manuel devant être acheminées. I_U doit être construite pour contenir les informations d'identification selon les procédures de l'infrastructure établie et peut également contenir d'autres informations, telles que l'utilisation prévue de la clé publique, le numéro de série du certificat manuel, et la période de validité du certificat manuel. La forme exacte de I_U dépend du format de certificat manuel spécifié pendant le procédé d'installation.
  - La chaîne d'octets MC_U, qui est le certificat manuel de U, doit être construite selon les procédures de l'infrastructure établie. MC_U doit contenir les chaînes d'octets I_U et WEU codées de manière réversible. La forme exacte de MC_U dépend du format de certificat manuel spécifié pendant le procédé d'installation.
- Sortie: MC_U, qui doit servir de certificat manuel de U fournie par CA.

H.10.3 Transformation de traitement de certificat manuel à cryptographie sur courbes elliptiques

V doit exécuter la transformation suivante pour obtenir la clé publique statique prétendue de U à partir du certificat manuel prétendu de U fourni par CA. V doit obtenir une copie authentique de l'identificateur de U et de CA.

- Entrée: Certificat manuel prétendu de U, MC_U, fourni par CA.
- Ingrédients: La transformation de traitement de certificat manuel utilise la primitive de validation de clé publique en SEC 1:2009, 3.2.2, et la primitive de validation de certificat manuel de l'infrastructure établie.
- Actions: V poursuit comme suit:
  - Vérifier le contenu de MCU selon l'infrastructure établie. Cela inclut de vérifier le contenu du certificat, tel que le nom du sujet et la période de validité. Si le nom du sujet est différent de U, produire une sortie invalide et s'arrêter.
  - Dériver I_U à partir de MCU, selon le format de certificat manuel spécifié pendant la procédure d'installation.
  - Dériver l'identificateur de CA à partir de I_U, selon le format de certificat spécifié pendant la procédure d'installation. Si l'identificateur du CA est inconnue de V, produire invalide et s'arrêter.
  - Dériver WE_U à partir de MCU, selon le format de certificat manuel spécifié pendant la procédure d'installation.
  - Convertir la chaîne d'octets WE_U en le point de courbe elliptique W_U tel que spécifié en SEC 1:2009, 2.3.4
  - Vérifier que W_U est une clé valide pour les paramètres D comme spécifié dans SEC 1:2009, 3.2.2. Si la primitive de validation rejette la clé, produire invalide et s'arrêter.
- Sortie: Si l'une quelconque des vérifications ci-dessous a échoué, produire invalide et s'arrêter; autrement, produire valide et accepter W_U comme la clé publique statique prétendue de U. (V peut accepter W_U comme U comme étant la clé publique statique authentique de U, à condition que U apporte à V la preuve de la connaissance de la clé privée correspondante w_U et à condition que V accepte que U soit la seule partie qui peut avoir accès à cette clé privée.)

Annexe I
(informative)

Modèles de définition

I.1 Modèle de type d'objet

Il convient que des objets normalisés et leurs identificateurs d'objets normalisés associés soient identifiés dans une table pour référence rapide, comme montré dans le Tableau I.1. Cela indique les informations exigées pour définir les types d'objets normalisées définis par la présente norme.

Tableau I.1 – Tableau des types d'objets normalisés

Définition de l'organisation:			
Nom du type d'objet normalisé **(transmission non attendue, taille non spécifiée – vérifier les limites DD)**	**Identificateur du type d'objet normalisé (non négatif)**	**Identificateur d'objet normalisé (non négatif), si applicable** **Utilisé pour objets obligatoires avec exactement une instance par appareil**	**Description de l'objet** **(transmission non attendue, taille non spécifiée – vérifier les limites DD)**
…	…	…	…

Les éléments de la table incluent:

- Le nom du type d'objet normalisé définit le nom de l'objet.
- L'identificateur du type d'objet normalisé non négatif est l'identificateur numérique normalisé non négatif du type d'objet; identifie uniquement ce type d'objet.
- L'identificateur d'objet normalisé, pour les types d'objets normalisés qui sont exigés par un appareil et qui peuvent être seulement instanciés une fois, représente l'identificateur numérique normalisé non négatif pour l'instance d'objet. Cet identificateur est commun à tous les appareils. Si 7 bits ne suffisent pas, le bit de poids fort du premier octet doit être mis et un autre octet doit être disponible pour étendre la valeur de l'identificateur.
- La description d'objet est une description du but et de l'intention de cet objet.

I.2 Modèles d'attributs d'objets normalisés

Le modèle montré dans le Tableau I.2 indique les informations nécessaires pour définir les attributs d'un objet normalisé.

Tableau I.2 – Modèle pour attributs d'objets normalisés

Nom du type d'objet normalisé:				
Identificateur du type d'objet normalisé:				
Définition de l'organisation:				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
ObjectIdentifier	Identificateur de clé	Identificateur unique pour l'objet	Type: Unsigned16.	N/A
			Classification: Statique	
			Plage valide: 1..32 767	
...	...	...	Type: ...	...
			Classification: ...	
			Accessibilité: ...	
			Valeur par défaut: ...	
			Plage valide: ...	
Réservée pour un usage futur	-	-	-	-

Les éléments de la table incluent:

- Le nom du type d'objet normalisé définit le nom de l'objet.
- L'identificateur du type d'objet normalisé est l'identificateur numérique normalisé non négatif du type d'objet; identifie uniquement ce type d'objet. La valeur de cet identificateur s'inscrit dans deux octets au maximum.
- L'organisation de définition est l'organisation définissant cet objet (par exemple, norme de base, extension définie normalisée de l'objet normalisé de base, profil spécifique à une industrie (et quelle industrie), groupe d'intérêt spécial (et quel groupe d'intérêt)), ou un fournisseur d'appareil.
- Le nom d'attribut définit le nom de l'attribut.
- L'identificateur d'attribut est un identificateur numérique normalisé de l'attribut. Tous les attributs d'un objet sont identifiés de façon univoque. Si 7 bits ne suffisent pas, le bit de poids fort du premier octet doit être mis et un autre octet doit être disponible pour étendre la valeur de l'identificateur.
- Description est la description de l'attribut.
- Type est le type de données de l'attribut. Si les données peuvent changer de taille (comme dans le cas d'un OctetString de taille variable ou d'un VisibleString de taille variable), le nombre maximal d'octets de données est indiqué.
- Classification est la classification des données ("constant", "static", "static-volatile", "dynamic", "non-cacheable") de l'attribut.
- Accessibilité est la manière dont l'attribut est accessible à distance (par exemple, read only, ou read/write)
- Valeur initiale par défaut spécifie la valeur initiale par défaut.
- Ensemble de valeurs valides spécifie l'ensemble valide de valeurs pour cet attribut.

I.3 Méthodes d'objets normalisés

Le modèle montré dans le Tableau I.3 indique les informations nécessaires pour décrire les méthodes d'un objet normalisé.

Tableau I.3 – Modèle pour méthodes d'objets normalisés

<table>
<tr><td colspan="5">Nom du type d'objet normalisé:</td></tr>
<tr><td colspan="5">Identificateur du type d'objet normalisé:</td></tr>
<tr><td colspan="5">Définition de l'organisation:</td></tr>
<tr><td>Nom de méthode</td><td>ID de méthode</td><td colspan="3">Description de la méthode</td></tr>
<tr><td rowspan="6"><nom de méthode></td><td colspan="4">Arguments d'entrée</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td>…</td><td>…</td><td>…</td><td>…</td></tr>
<tr><td colspan="4">Arguments de sortie</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td>…</td><td>…</td><td>…</td><td>…</td></tr>
</table>

Les éléments de la table incluent:

- Le nom du type d'objet normalisé définit le nom de l'objet.
- L'identificateur du type d'objet normalisé est l'identificateur numérique normalisé non négatif du type d'objet; identifie uniquement ce type d'objet. La valeur de cet identificateur s'inscrit dans deux octets au maximum.
- L'organisation de définition est l'organisation définissant cet objet (par exemple, norme de base, extension définie normalisée de l'objet normalisé de base, profil spécifique à une industrie (et quelle industrie), groupe d'intérêt spécial (et quel groupe d'intérêt)).
- Le nom de méthode est le nom de la méthode.
- L'ID de méthode est l'identification numérique de la méthode. Toutes les méthodes d'un objet auront des identificateurs de méthode uniques. Si 7 bits ne suffisent pas, le bit de poids fort du premier octet doit être mis et un autre octet doit être disponible pour étendre la valeur de l'identificateur.
- Description de méthode est la description de la méthode.
- La liste de paramètres d'entrée et de leurs types de données est une liste des paramètres d'entrée, de leur type et de leur taille (si elle n'est pas explicitement discernable du type), et une description d'utilisation (comment ils sont utilisés lorsqu'ils sont envoyés sur une invocation de méthode). Il convient de les énumérer dans l'ordre d'émission.

NOTE 1 Pour simplifier, tous les paramètres sont spécifiés. S'il y a des situations où les paramètres varient, des méthodes séparées sont appropriées pour prendre en charge chaque classe de variance.

- La liste de paramètres de sortie et de leurs types de données est une liste des paramètres de sortie, de leur type et de leur taille (si elle n'est pas explicitement discernable du type), et une description d'utilisation (comment ils sont utilisés lorsqu'ils sont envoyés sur une invocation de méthode). Il convient de les énumérer dans l'ordre d'émission.

NOTE 2 Voir NOTE 1.

- Description de comportement décrit le comportement de l'objet lorsque cette méthode est invoquée.

I.4 Modèles de rapports d'alerte d'objets normalisés

Le modèle montré dans le Tableau I.4 indique les informations nécessaires pour décrire le comportement de rapports d'alertes d'un objet normalisé.

Tableau I.4 – Modèles pour rapports d'alerte d'objets normalisés

Nom du type d'objet normalisé:					
Identificateur du type d'objet normalisé:					
Définition de l'organisation:					
Description de l'alerte:					
Classe d'alertes (Enumerated: alarme ou événement)	**Catégorie d'alertes (Enumerated: diagnostic d'appareil, diagnostic de comm., sécurité ou processus)**	**Type d'alerte (Enumerated: en fonction de la catégorie d'alertes)**	**Alert priority**	**Type de données de la valeur**	**Description de la valeur incluse avec l'alerte**
<nom d'alerte>	...	...	...	Type: ...	...
				Valeur par défaut: ...	...
				Plage valide: ...	...

Les éléments de la table incluent:

- Le nom du type d'objet normalisé définit le nom de l'objet.
- L'identificateur du type d'objet normalisé est l'identificateur numérique normalisé du type d'objet qui identifie de manière unique ce(s) type(s) d'objet qui peut(peuvent) rapporter cette alerte. La valeur de cet identificateur s'inscrit dans deux octets au maximum.
- L'organisation de définition est l'organisation définissant cet objet (par exemple, norme de base, profil spécifique à une industrie (et quelle industrie), groupe d'intérêt spécial (et quel groupe d'intérêt)).
- La description de l'alerte décrit la signification sémantique de l'alerte.
- Classe d'alertes indique s'il s'agit d'un type d'alerte événement (sans états) ou alarme (orientée état).
- Catégorie d'alerte indique s'il s'agit d'une alerte relative à un appareil (par exemple, diagnostic spécifique à un appareil), relative à une communication, relative à la sécurité, ou relative à un processus. Une seule catégorie s'applique. La sélection du meilleur ajustement pour une alerte peut avoir besoin d'être débattue afin d'être mieux établie.
- Le type alerte dépend de la catégorie alerte. Voir le modèle de rapports d'alertes en 12.8 pour plus de détails.
- Priorité d'alerte est la priorité de l'alerte.
- Valeur et taille représentent la taille et la valeur incluses dans le rapport d'alerte.
- La description de la valeur incluse dans le rapport d'alerte est la description de la valeur, si une valeur est incluse dans le rapport d'alerte (par exemple, pour une alarme de processus qui est une alarme "high", il peut s'agir de variable de processus (PV)).
- Accessibilité définit si l'attribut est lisible, inscriptible ou les deux.
- La valeur initiale par défaut indique la valeur initiale par défaut de l'attribut.
- Description de l'ensemble de valeurs décrit l'ensemble de valeurs qui peuvent être prises par cet attribut.
- Description du comportement décrit le comportement de cet attribut (par exemple, quand une valeur particulière est écrite ou en cas de condition d'erreur). Des restrictions à l'utilisation (par exemple, il convient que les opérateurs n'écrivent pas dans cet attribut) peuvent être notées ici.

I.5 Définition de structures de données

Le modèle pour décrire les structures de données qui sont utilisées pour définir des types de données spéciaux est donné dans le Tableau I.5.

Tableau I.5 – Modèle pour structures de données

<table>
<tr><th colspan="3">Nom du type de données normalisé:</th></tr>
<tr><th colspan="3">Définition de l'organisation:</th></tr>
<tr><th>Nom de l'élément</th><th>Identificateur de l'élément</th><th>Type de l'élément scalaire</th></tr>
<tr><td rowspan="6">...</td><td rowspan="6">...</td><td>Type: ...</td></tr>
<tr><td>Taille: ...</td></tr>
<tr><td>Classification: ...</td></tr>
<tr><td>Accessibilité: ...</td></tr>
<tr><td>Valeur par défaut: ...</td></tr>
<tr><td>Plage valide: ...</td></tr>
</table>

Annexe J
(informative)

Opérations sur les attributs

J.1 Opérations sur les attributs

J.1.1 Généralités

La classification et l'accessibilité d'attribut dictent les opérations autorisées sur un attribut donné. La classification et l'accessibilité d'attribut sont décrites en 12.6.

J.1.2 Classification d'attributs

Pour un débat relatif à la classification d'attribut, voir 12.6.3.

J.1.3 Récupération, positionnement et réinitialisation d'attributs

J.1.3.1 Généralités

Les attributs définis dans les objets de gestion peuvent être accessibles en utilisant les services normalisés de lecture ou d'écriture fournis par l'ASL. De telles opérations permettent la configuration des couches, ainsi que la surveillance de leur statut. Elles peuvent être utilisées pour récupérer, établir/modifier, et réinitialiser les valeurs des attributs. Les primitives de service pour ces services ainsi que les codes énumérés de retour d'informations de service, sont donnés à l'Article 12.

Des attributs peuvent être réinitialisés en utilisant le service write en écrivant la valeur par défaut dans l'attribut pertinent. Si un attribut reset est défini pour un objet de gestion, il peut être utilisé pour réinitialiser tous les attributs dans l'objet de gestion en question qui appartiennent à certaines classes d'attributs.

Des méthodes plus complexes peuvent être définies s'il y a lieu, mais seulement si les résultats équivalents ne peuvent pas être obtenus en utilisant les services read/write plus directs. Une méthode complexe peut être justifiée, par exemple, pour remplacer une séquence de transactions de communication afin d'économiser de l'énergie. Une méthode complexe peut également être justifiée lorsque des questions de synchronisation peuvent résulter si des actions individuelles sont utilisées plutôt qu'un ensemble de transactions atomiques.

J.1.3.2 Opérations programmées pour activer un basculement synchronisé

Le modèle de méthode générique Scheduled_Write fourni dans le Tableau J.1 peut être utilisé pour définir une méthode pour écrire une valeur dans un attribut à un temps TAI programmé. Il peut également être utilisé pour réinitialiser un attribut à sa valeur par défaut à un temps TAI programmé.

Tableau J.1 – Modèle de méthode Scheduled_Write

<table>
<tr><th>Nom de méthode</th><th>ID de méthode</th><th colspan="3">Description de la méthode</th></tr>
<tr><td rowspan="9">Scheduled_Write</td><td><donné dans la définition de l'objet de gestion></td><td colspan="3">Méthode pour écrire une valeur dans un attribut indiqué à un temps TAI indiqué</td></tr>
<tr><td colspan="4">Arguments d'entrée</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td>1</td><td>Attribute_ID</td><td>Data type: Unsigned16 <donné dans la définition de l'objet de gestion></td><td>L'ID d'attribut dans l'objet de gestion auquel la présente méthode est appliquée</td></tr>
<tr><td>2</td><td>Scheduled_TAI_Time</td><td>Data type: TAITimeRounded</td><td>Temps TAI auquel il convient que la valeur soit écrite dans l'attribut</td></tr>
<tr><td>3</td><td>Valeur</td><td>Data type: <donné dans la définition de l'objet de gestion></td><td>La valeur qui a besoin d'être écrite dans l'attribut</td></tr>
<tr><td colspan="4">Arguments de sortie</td></tr>
<tr><td>Numéro de l'argument</td><td>Nom de l'argument</td><td>Type de l'argument (type et taille des données)</td><td>Description de l'argument</td></tr>
<tr><td colspan="4">Aucun</td></tr>
</table>

Les codes de retour d'informations de services en 12.17.4.2.2 sont censés être utilisés pour indiquer si l'exécution de la méthode a réussi ou pas. Si elle n'a pas réussi, ce code fournit des informations indiquant pourquoi elle n'a pas réussi.

J.1.4 Récupération et positionnement d'attributs structurés

Les modèles de méthode génériques Read_Row et Write_Row donnés dans le Tableau J.2 et dans le Tableau J.3 peuvent être utilisés pour définir les méthodes qui récupèrent et établissent/modifient les valeurs d'attributs structurés. Lorsque l'attribut structuré est visualisé sous la forme d'une table d'informations, ces méthodes permettent l'accès à une rangée particulière en fonction d'une ou plusieurs valeurs de champ indice unique. Il est supposé que chaque table a au moins un champ indice unique. Le champ indice peut être soit un seul élément, soit la concaténation de quelques éléments dans la rangée.

L'argument d'entrée Scheduled_TAI_Time dans le modèle de méthode Write_Row permet l'opération programmée pour une rangée particulière de l'attribut structuré. Une valeur de 0 pour cet argument indique une opération write immédiate.

Tableau J.2 – Modèle de méthode Read_Row

Nom de méthode	ID de méthode	Description de la méthode		
Read_Row	<donné dans la définition de l'objet de gestion>	Méthode pour lire la valeur d'une seule rangée d'un attribut structuré dont les données sont visualisées sous la forme d'un tableau d'informations		
	Arguments d'entrée			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	Attribute_ID	Data type: Unsigned16 <donné dans la définition de l'objet de gestion>	L'ID d'attribut dans l'objet de gestion auquel la présente méthode est appliquée
	2	Index_1	Type de données du premier champ d'index de l'attribut structuré <donné dans la définition de l'objet de gestion>	Le premier champ d'index dans l'attribut structuré pour accéder à une rangée particulière
	n+1	Index_n	Type de données du $n^{ème}$ champ d'index de l'attribut structuré <donné dans la définition de l'objet de gestion>	Le $n^{ème}$ champ d'index dans l'attribut pour accéder à une rangée particulière
	Arguments de sortie			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	Data_Value	Data type: <donné dans la définition de l'objet de gestion>	Une chaîne d'octet qui contient la valeur de données

Tableau J.3 – Modèle de méthode Write_Row

Nom de méthode	ID de méthode	Description de la méthode		
Write_Row	<donné dans la définition de l'objet de gestion>	Méthode pour définir/modifier la valeur d'une seule rangée d'un attribut OctetString indexé dont les données sont visualisées sous la forme d'un tableau d'informations		
	Arguments d'entrée			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	Attribute_ID	Data type: Unsigned16 <donné dans la définition de l'objet de gestion>	L'ID d'attribut dans l'objet de gestion auquel la présente méthode est appliquée
	2	Scheduled_TAI_Time	Data type: TAITimeRounded	Temps TAI auquel il convient que la valeur soit écrite dans la rangée de l'attribut structuré
	3	Index_1	Type de données du premier champ d'index de l'attribut structuré <donné dans la définition de l'objet de gestion>	Le premier champ d'index dans l'attribut structuré pour accéder à une rangée particulière
	N+2	Index_n	Type de données du $n^{ème}$ champ d'index de l'attribut structuré <donné dans la définition de l'objet de gestion>	Le $n^{ème}$ champ d'index dans l'attribut pour accéder à une rangée particulière
	N+3	Data_Value	Data type: <donné dans la définition de l'objet de gestion>	Une chaîne d'octet qui contient la valeur de données
	Arguments de sortie			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	Aucun			

Les codes de retour d'informations de services en 12.17.4.2.2 sont censés être utilisés pour indiquer si l'exécution de la méthode a réussi ou pas. Si elle n'a pas réussi, ce code fournit des informations indiquant pourquoi elle n'a pas réussi.

Une méthode basée sur le modèle Write_Row peut également être utilisée pour créer une nouvelle rangée dans l'attribut structuré si le(s) champ(s) indice fourni(s) dans l'(les) argument(s) d'entrée n'existe(nt) pas.

J.1.5 Réinitialisation des valeurs d'attributs structurés

Pour un attribut structuré, le modèle de méthode générique Reset_Row donné dans Tableau J.4 peut être utilisé pour définir des méthodes qui réinitialisent/effacent certaines valeurs dans l'attribut structuré. L'argument d'entrée Scheduled_TAI_Time dans cette méthode permet une opération reset programmée. Une valeur de 0 pour cet argument indique une opération reset immédiate.

Tableau J.4 – Modèle de méthode Reset_Row

Nom de méthode	**ID de méthode**	**Description de la méthode**		
Reset_Row	<donné dans la définition de l'objet de gestion>	Méthode pour réinitialiser une seule rangée d'un attribut structuré dont les données sont visualisées sous la forme d'un tableau d'informations		
	Arguments d'entrée			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	Attribute_ID	Data type: Unsigned16 <donné dans la définition de l'objet de gestion>	L'ID d'attribut dans l'objet de gestion auquel la présente méthode est appliquée
	2	Scheduled_TAI_Time	Data type: TAITimeRounded	Temps TAI auquel il convient que l'attribut structuré soit réinitialisé
	3	Index_1	Type de données du premier champ d'index de l'attribut structuré <donné dans la définition de l'objet de gestion>	Le premier champ d'index dans l'attribut structuré pour accéder à une rangée particulière
	n+2	Index_n	Type de données du $n^{ème}$ champ d'index de l'attribut structuré <donné dans la définition de l'objet de gestion>	Le $n^{ème}$ champ d'index dans l'attribut pour accéder à une rangée particulière
	Arguments de sortie			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	Aucun			

Les codes de retour d'informations de services en 12.17.4.2.2 sont censés être utilisés pour indiquer si l'exécution de la méthode a réussi ou pas. Si elle n'a pas réussi, ce code fournit des informations indiquant pourquoi elle n'a pas réussi.

J.1.6 Suppression de valeurs d'attributs structurés

Le modèle de méthode générique Delete_Row décrit dans Tableau J.5 peut être utilisé pour définir les méthodes qui suppriment les valeurs d'attributs structurés. L'argument d'entrée

Scheduled_TAI_Time dans cette méthode permet une opération delete (suppression) programmée. Une valeur de 0 pour cet argument indique une opération delete immédiate.

Tableau J.5 – Modèle de méthode Delete_Row

Nom de méthode	ID de méthode	Description de la méthode		
Delete_Row	<donné dans la définition de l'objet de gestion>	Méthode pour supprimer une seule rangée d'un attribut structuré dont les données sont visualisées sous la forme d'un tableau d'informations		
	Arguments d'entrée			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	1	Attribute_ID	Data type: <Unsigned16 donné dans la définition de l'objet de gestion>	L'ID d'attribut dans l'objet de gestion auquel la présente méthode est appliquée
	2	Scheduled_TAI_Time	Data type: TAITimeRounded	Temps TAI auquel il convient que l'attribut structuré soit supprimé
	3	Index_1	Type de données du premier champ d'index de l'attribut structuré <donné dans la définition de l'objet de gestion>	Le premier champ d'index dans l'attribut structuré pour accéder à une rangée particulière
	n+2	Index_n	Type de données du $n^{ème}$ champ d'index de l'attribut structuré <donné dans la définition de l'objet de gestion>	Le $n^{ème}$ champ d'index dans l'attribut pour accéder à une rangée particulière
	Arguments de sortie			
	Numéro de l'argument	**Nom de l'argument**	**Type de l'argument (type et taille des données)**	**Description de l'argument**
	Aucun			

Les codes de retour d'informations de services en 12.17.4.2.2 sont censés être utilisés pour indiquer si l'exécution de la méthode a réussi ou pas. Si elle n'a pas réussi, ce code fournit des informations indiquant pourquoi elle n'a pas réussi.

J.2 Basculement synchronisé

Une capacité synchronisée de basculement est nécessaire pour certains attributs et attributs structurés qui représentent les informations de gestion. Pour un tel attribut, des mises à jour pour la valeur d'attribut peuvent être programmées en indiquant les informations relatives au temps de basculement TAI; cette opération peut être accomplie en utilisant une des méthodes définies ci-dessus. De telles mises à jour sont envoyées à l'objet de gestion pour lequel cet attribut est défini. L'objet de gestion valide immédiatement si, oui ou non, le basculement est faisable, et, si faisable, s'arrange pour que le basculement se produise au temps programmé.

Annexe K
(normative)

Types d'objets normalisés

L'Annexe K spécifie les types d'objets normalisés définis par la présente norme. Chaque type d'objet a trois éléments d'informations pour l'identifier:

- un identificateur de type d'objet normalisé correspondant qui identifie le type d'objet de basse défini normalisé (exemple: entrée analogique);
- un identificateur de sous-type normalisé d'objet correspondant qui identifie le sous-type d'objet normalisé d'un type de base normalisé (exemple: entrée analogique spécialisée pour la température); et
- un identificateur de sous-type de fournisseur correspondant qui identifie un sous-type spécifique à un fournisseur soit d'un objet de base normalisé, soit d'un sous-type normalisé.

Les objets de base normalisés doivent avoir leur valeur d'identificateur de sous-type d'objet égale à zéro (0) et leur identificateur de sous-type de fournisseur égal à zéro (0).

Une plus récente version de la présente norme qui trouve nécessaire d'étendre la définition du type d'objet de base selon la présente norme peut maintenir la valeur d'identificateur d'objet normalisé et la valeur de sous-type de zéro (0). Cela est autorisé, car le DMO contient un attribut pour représenter la version de la norme utilisée par l'appareil, qui peut être ainsi utilisé pour établir la structure utilisée de type d'objet de base.

Les profils d'industrie IEC62734 peuvent définir un sous-type d'objet normalisé comme objet normalisé. Le faire crée ainsi un sous-type de profil normalisé. La présente norme fournit une plage de 1..255 pour représenter tous ces sous-types d'objets normalisés à travers tous les profils.

Les fournisseurs peuvent également sous-typer soit un objet de base normalisé, soit un objet de sous-type normalisé. La présente norme fournit une gamme de 1..255 pour le sous-typage spécifique à un fournisseur.

Le sous-typage d'objet a lieu quand:

- un ou plusieurs attributs sont ajoutés au type de base;
- une ou plusieurs méthodes sont ajoutées au type de base;
- une ou plusieurs alertes sont ajoutées au type de base; ou
- toute combinaison de ce qui précède.

Les exemples d'identification d'objet avec sous-typage suivent:

- Objet de bas normalisé d'entrée analogique:
  - identificateur de type d'objet = 99,
  - identificateur de sous-type normalisé d'objet = 0,
  - identificateur de sous-type de fournisseur = 0.
- Objet de sous-type de température d'entrée analogique:
  - identificateur de type d'objet = 99,
  - identificateur de sous-type normalisé d'objet = (la présente norme définit (l'équipe de profils de la présente norme), plage 1..255),
  - identificateur de sous-type de fournisseur = 0.

- Objet d'entrée analogique spécifique à un fournisseur:
  - identificateur de type normalisé d'objet = 99,
  - identificateur de sous-type normalisé d'objet = 0,
  - identificateur de sous-type spécifique à un fournisseur = (le fournisseur définit, plage 1..255).
- Objet température d'entrée analogique spécifique à un fournisseur:
  - identificateur de type normalisé d'objet = 99,
  - identificateur de sous-type normalisé d'objet = n,
  - identificateur de sous-type spécifique à un fournisseur = (le fournisseur définit, plage 1..255).

Le Tableau K.1 spécifie les types d'objets normalisés.

Tableau K.1 – Types d'objets normalisés

Type d'objet	Identificateur du type d'objet normalisé (1 octet)	Identificateur du sous-type d'industrie d'objet normalisé (1 octet)	Restrictions d'identificateur d'objet
Types d'objets disponibles pour tous les UAP			
Objet nul	0	0	Réservé
Objet de gestion d'UAP	1	0	Cet objet est exigé pour tous les UAP, mais n'est pas exigé pour le DMAP
Object AlertReceiving	2	0	
Objet UploadDownload	3	0	
Objet Concentrator	4	0	
Objet Dispersion	5	0	
Objet Tunnel	6	0	
Objet d'interface	7	0	
Réservé pour usage par la présente norme pour les objets UAP normalisés	8..50	0	Réservé pour les définitions d'objets normalisés dans le futur pour les objets indépendants du profile
Réservé pour usage par la présente norme	51..95	0	Types spécifiques à l'industrie
Types d'objets d'industrie de contrôle du processus			
Entrée analogique	99	0	Entrée analogique
Sortie analogique	98	0	Sortie analogique
Entrée binaire	97	0	Entrée binaire
Sortie binaire	96	0	Sortie binaire
Types d'objets DMAP			
DMAP: Device management object	127	0	Cet objet facilite la gestion des fonctions générales à tout l'appareil de l'appareil
DMAP: Alert reporting management object (ARMO)	126	0	Cet objet facilite la gestion des fonctions de rapport d'alerte de l'appareil
DMAP: Device security management object (DSMO)	125	0	Cet objet facilite la gestion des fonctions de sécurité de l'appareil

Type d'objet	Identificateur du type d'objet normalisé (1 octet)	Identificateur du sous-type d'industrie d'objet normalisé (1 octet)	Restrictions d'identificateur d'objet
DMAP: DL management object (DLMO)	124	0	Cet objet facilite la gestion des DL de l'appareil
DMAP: NL management object (NLMO)	123	0	Cet objet facilite la gestion des NL de l'appareil
DMAP: TL management object (TLMO)	122	0	Cet objet facilite la gestion des TL de l'appareil
DMAP: Application sublayer management object (ASLMO)	121	0	Cet objet facilite la gestion de la sous-couche d'application de l'appareil
DMAP: Device provisioning object (DPO)	120	0	Cet objet facilite la configuration de l'appareil avant qu'il ne rejoigne le réseau
DMAP: Health reports concentrator object (HRCO)	128	0	Cet objet facilite la publication périodique des rapports de santé d'appareil au gestionnaire système
Objets de gestion normalisés	119..114	0	
System time service object (STSO)	100	0	Cet objet facilite la gestion des informations de temps dans tout le système
Directory service object (DSO)	101	0	Cet objet facilite la gestion des adresses de tous les appareils existants dans le réseau
System configuration object (SCO)	102	0	Cet objet facilite la configuration du système, y compris l'établissement, la modification et la fin de contrat
Device management service object (DMSO)	103	0	Cet objet facilite l'attachement d'appareil, la fermeture d'appareil et la configuration d'appareil
System monitoring object (SMO)	104	0	Cet objet facilite la surveillance des performances du système
Proxy security management object (PSMO)	105	0	Cet objet agit en tant que proxy pour le gestionnaire de sécurité
Device provisioning service object (DPSO)	106	0	Cet objet facilite la configuration d'appareil
Objets de gestion de système normalisés	107..113	0	Réservé pour les définitions de type d'objet de gestion normalisés. Voir l'Article 6 pour plus de détails
Types définis par le fournisseur			
Objets définis par le fournisseur	129..255	0	Réservé pour usage par les metteurs en œuvre

Le Tableau K.2 spécifie les instances d'objets normalisés.

Tableau K.2 – Instances d'objets normalisés

Type d'objet	Identificateur du type d'objet normalisé (1 octet)	Identificateur du sous-type d'industrie d'objet normalisé (1 octet)	Identificateur de l'objet normalisé (1 octet)	Restrictions d'identificateur d'objet
Types d'objets disponibles pour tous les UAP				
Objet nul	0	0	0	Réservé
Objet de gestion d'UAP	1	0	1	Cet objet est exigé pour tous les UAP, mais n'est pas exigé pour le DMAP
Objet UploadDownload	3	0	2	Pour utilisation lors d'une mise à niveau UAP uniquement
Types d'objets d'industrie de contrôle du processus				
N/A				
Types d'objets DMAP				
DMAP: Device management object (DMO)	127	0	1	Cet objet facilite la gestion des fonctions générales à tout l'appareil de l'appareil
DMAP: Alert reporting management object (ARMO)	126	0	2	Cet objet facilite la gestion des fonctions de rapport d'alerte de l'appareil
DMAP: Device security management object (DSMO)	125	0	3	Cet objet facilite la gestion des fonctions de sécurité de l'appareil
DMAP: DL management object (DLMO)	124	0	4	Cet objet facilite la gestion des DL de l'appareil
DMAP: NL management object (NLMO)	123	0	5	Cet objet facilite la gestion des NL de l'appareil
DMAP: TL management object (TLMO)	122	0	6	Cet objet facilite la gestion des TL de l'appareil
DMAP: Application sublayer management object (ASLMO)	121	0	7	Cet objet facilite la gestion de la sous-couche d'application de l'appareil
DMAP: Upload/download object (UDO)	3	0	8	Cet objet facilite la gestion des fonctions de téléchargement montant/descendant de l'appareil
DMAP:Device provisioning object (DPO)	120	0	9	Cet objet facilite la configuration de l'appareil avant qu'il ne rejoigne le réseau
DMAP: Health reports concentrator object (HRCO)	128	0	10	Cet objet facilite la publication périodique des rapports de santé d'appareil au gestionnaire système

Type d'objet	Identificateur du type d'objet normalisé (1 octet)	Identificateur du sous-type d'industrie d'objet normalisé (1 octet)	Identificateur de l'objet normalisé (1 octet)	Restrictions d'identificateur d'objet
Types normalisés AP de gestion système				
System time service object (STSO)	100	0	1	Cet objet facilite la gestion des informations de temps dans tout le système
Directory service object (DSO)	101	0	2	Cet objet facilite la gestion des adresses de tous les appareils existants dans le réseau
System configuration object (SCO)	102	0	3	Cet objet facilite la configuration du système, y compris l'établissement, la modification et la fin de contrat
Device management service object (DMSO)	103	0	4	Cet objet facilite l'attachement d'appareil, la fermeture d'appareil et la configuration d'appareil
System monitoring object (SMO)	104	0	5	Cet objet facilite la surveillance des performances du système
Proxy security management object (PSMO)	105	0	6	Cet objet agit en tant que proxy pour le gestionnaire de sécurité
Upload/download object (UDO)	3	0	7	Cet objet facilite le téléchargement de firmware/de données vers les appareils et le chargement de données à partir des appareils
Alert-receiving object (ARO)	2	0	8	Cet objet reçoit toutes les alertes destinées au gestionnaire de système
Device provisioning service object (DPSO)	106	0	9	Cet objet facilite la configuration d'appareil
Health reports concentrator object (HRCO)	4	0	10	Cet objet facilite la publication périodique des rapports de santé d'appareil au gestionnaire système
Types définis par le fournisseur				
…	…	…	…	…

Annexe L
(informative)

Types de données normalisés

Le Tableau L.1 spécifie les identificateurs de types de données normalisés pour les types de données normalisés. Des types de données normalisés sont définis pour les constructions qui sont accessibles en utilisant des services d'ASL, tels que read et write.

NOTE 1 Il est possible que les structures de données ne soient pas directement accessibles en utilisant des services ASL, par exemple, une structure de données qui est utilisée comme paramètre d'une méthode, mais qui n'est pas exposée comme un attribut d'objet accessible à l'ASL.

NOTE 2 Beaucoup d'identificateurs de types dans cette tableau sont basés sur des identificateurs de types utilisés dans une norme IEC existante.

Tableau L.1 – Types de données normalisés

Type de données	Identificateur de type (Unsigned16)	Taille (octets)
Types réservés		
Non valide (type non spécifié)	0	0
Types de structure de données normalisées AP		
Communication association endpoint (point d'extrémité d'association de communication)	468	Voir Tableau 265
ObjectAttributeIndexAndSize	469	Voir Tableau 264
Communication contract data (données de contrat de communication)	470	Voir Tableau 266
Alert communication endpoint (point d'extrémité de communication d'alertes)	471	Voir Tableau 267
ObjectIDandType	472	Voir Tableau 271
Unscheduled correspondent (correspondant non programmé)	473	Voir Tableau 272
Echelonnement de contrôle de processus		
Process control value and status for analog value	65	Voir Tableau 300
Process control value and status for binary value	66	Voir Tableau 301
Process control scaling	68	Voir Tableau 304
Process control mode (mode contrôle de processus)	69	Voir Tableau 302
Types de descripteur d'alerte		
Descripteur de rapports d'alarmes de contrôle de processus pour analogique avec une seule condition de référence	498	Voir Tableau 270
Alert report descriptor (descripteur de rapports d'alertes) (aussi utilisé pour les alarmes binaires de contrôle de processus)	499	Voir Tableau 269

Type de données	Identificateur de type (Unsigned16)	Taille (octets)
Types de gestion/communication généraux		
Contract_Data	401	Voir Tableau 30
Address_Translation_Row	402	Voir Tableau 14
New_Device_Contract_Response	405	Voir Tableau 31
Metadata_attribute	406	Voir Tableau 2
Security_Sym_Join_Request	410	Voir Tableau 62
Security_Sym_Join_Response	411	Voir Tableau 63
Security_Sym_Confirm	412	Voir Tableau 66
Security_Pub_Join_Request	415	Voir Tableau 70
Security_Pub_Join_Response	416	Voir Tableau 70
Security_Pub_Confirm_Request	417	Voir Tableau 72
Security_Pub_Confirm_Response	418	Voir Tableau 72
Security_New_Session_Request	420	Voir Tableau 81
Security_New_Session_Response	421	Voir Tableau 82
Security_Key_and_Policies	422	Voir Tableau 84
Security_Key_Update_Status	423	Voir Tableau 85
DPSOWhiteListTbl	440	Voir Tableau 372
NLContractTbl	441	Voir Tableau 207
NLRouteTbl	442	Voir Tableau 208
NLATTbl	443	Voir Tableau 209

Annexe M
(normative)

Identification de protocoles de bus de terrain hérités et tunnellisés

Le Tableau M.1 énumère les valeurs d'identification du protocole Unsigned8 actuellement définies pour tunnelliser des protocoles de bus de terrain câblés hérités par l'intermédiaire de l'objet tunnel.

Tableau M.1 – Identification de protocole de bus de terrain hérités tunnellisés

Valeur	Protocole
0	Aucun (exigé)
1	HART (voir IEC 61158)
2	FF-H1 (voir IEC 61158)
3	Modbus/RTU (voir IEC 61158)
4	PROFIBUS PA (voir IEC 61158)
5	CIP (voir IEC 61158)
6..255	<réservé>

NOTE Ces valeurs d'identification de protocole ont été isolées dans l'Annexe M afin de faciliter la maintenance.

Il convient que la valeur 0 pour None soit préservée ou la fonctionnalité de tunnel sera entravée.

Annexe N
(informative)

Tunnellisation et mapping d'objets natifs

N.1 Vue d'ensemble

La tunnellisation implique l'échange de PDU d'un protocole en utilisant un second protocole. Le plus souvent, ces PDU sont des PDU d'application, mais les PDU de couche inférieure peuvent aussi être échangées. La PDU est encapsulée dans le second protocole en un nœud d'origine et envoyée par le réseau à un nœud final. Avec la tunnellisation, ce qui entre dans une extrémité sort de l'autre extrémité, ni plus, ni moins.

La communication d'application de protocole étranger (FPAC) est un mécanisme d'échange de PDU plus sophistiqué. Elle implique l'utilisation de mécanismes complémentaires, y compris le placement en cache, la compression, la conversion d'adresses, et le proxy. En ce qui concerne l'application, les mêmes PDU sont encore échangées entre le nœud d'origine et le nœud de terminaison comme avec la tunnellisation. La différence est que les mécanismes complémentaires agissent pour améliorer l'efficacité énergétique et la réactivité du système hôte.

N.2 Tunnellisation

La tunnellisation transporte des messages in extenso entre les points d'extrémité d'un tunnel. La présente norme fournit la tunnellisation qui utilise l'échange C/S non tamponné de PDU étrangères entre exactement deux points d'extrémité préconfigurés du tunnel. Aucune interprétation du contenu de la PDU n'est exigée. Pour la plupart des protocoles hérités, cette méthode ne sera pas d'un haut rendement énergétique et certains protocoles peuvent ne pas fonctionner correctement en raison de temps de réponse variables ou prolongés associés aux appareils en sommeil. Indépendamment des défauts, dans de nombreux cas, ce sera la méthode la plus indiquée pour adapter des appareils et systèmes existants à la présente norme.

Une extension de tunnellisation interprète l'adressage au sein des PDU étrangères pour permettre l'échange dynamique de PDU étrangères avec plusieurs points d'extrémité.

N.3 Communication d'application de protocole étranger

La tunnellisation n'est pas un mécanisme approprié pour la plupart des applications de liaison sans fil de faible puissance. Il est habituellement nécessaire de réduire au maximum le surdébit de PhPDU et le nombre de transactions afin de conserver l'énergie emmagasinée dans des batteries ou de fonctionner dans les limites d'un budget de puissance des méthodes de récupération. En outre, les protocoles étrangers ont souvent un besoin de réponse rapide afin d'éviter des temporisations intégrées. Les appareils en fonctionnement sans fil de faible puissance sont le plus souvent en mode de sommeil et ne peuvent donc pas répondre immédiatement.

La FPAC augmente l'efficacité énergétique et traite des questions potentielles de temporisation en utilisant le transfert à changement d'état et le placement en cache pour éliminer le transfert redondant. Des améliorations de l'efficacité énergétique et des performances sont réalisées en plaçant en cache les informations dans la passerelle, en transférant les informations vers la passerelle seulement lorsqu'elles changent, et en fournissant un mécanisme de signaux de présence ("heartbeat") pour l'intégrité. Cela réduit au maximum les transferts initiés par les appareils d'extrémité (à savoir publications périodiques), et réduit également au maximum des transferts initiés par la liaison de

communication étrangère (à savoir accès de plusieurs maîtres par la passerelle. En outre, cette méthode peut traiter des exigences de temporisation de protocoles étrangers. En comparaison à la tunnellisation, un effort supplémentaire est nécessaire pour convertir le protocole étranger.

La présente norme fournit la prise en charge de la FPAC qui réduit au maximum le surdébit de PhPDU en utilisant une combinaison de techniques:

- L'encapsulation est limitée à une seule encapsulation. Les convertisseurs de protocoles fournissent l'encapsulation complémentaire à travers les liaisons étrangères selon les besoins.
- L'encapsulation est réalisée à travers un accord de configuration en transportant le protocole étranger au sein du protocole défini par la présente norme, plutôt qu'en transportant des en-têtes de protocoles supplémentaires. Le mapping se produit comme suit:
  - Relations prises en charge par le transport (P/S et C/S).
  - Adresses étrangères et adresses natives.
  - Champs taille et champs intégrité.
- La présente norme fournit un format natif de service d'application pour l'échange de messages. Les protocoles étrangers ont leurs propres formats de service et leurs propres protocoles d'échange de messages. L'objet de tunnel permet le transfert d'APDU étrangères sans surdébit extérieur imposé par le format natif de service d'application.

La présente norme fournit la prise en charge de la FPAC qui réduit au maximum le surdébit de transactions en utilisant les techniques suivantes:

- Mécanismes de placement en cache de tampon distribués pour réduire au maximum le transfert redondant des données inchangées entre les passerelles et les appareils d'extrémité.
- Mécanismes de transferts périodiques, de changement d'états (CoSt), et apériodiques.
- Temporisateurs chiens de garde pour surveiller la disponibilité de points d'extrémité et de voies de communication et assurer la qualité de données.

La présente norme fournit la prise en charge de la FPAC qui améliore les performances de temporisation d'accès d'appareils de protocoles étrangers (et réduit au maximum les transactions inutiles) par la mise à disposition d'informations d'appareil tamponnées par une interface côté haut d'une passerelle.

NOTE Le changement d'état (CoSt) est distinct de la classe de service (CoS) telle que définie par l'IEEE 802.1Q.

N.4 Mapping d'objets natifs

La présente norme prend en charge un format natif d'objets et des services de messagerie. Des objets spécifiques à l'automation peuvent être utilisés pour prendre en charge la conversion de protocole en utilisant ces objets pour accomplir un mapping du protocole étranger dans ces objets et leur messagerie. En comparaison à la tunnellisation et aux méthodes de la FPAC, un effort supplémentaire est nécessaire pour convertir le protocole étranger.

N.5 Compris entre tunnellisation et mappage d'objets natifs

Le mapping d'objets natifs a un avantage unique dans la capacité d'établir un seul appareil d'extrémité conforme à une norme avec plusieurs protocoles étrangers. Cela est notamment attrayant pour de nouveaux appareils.

La tunnellisation et la FPAC ont un avantage en la simplicité pour adapter des appareils câblés d'automation par l'intermédiaire d'un adaptateur. Très peu, voire pas du tout, de conversion peut être nécessaire à l'autre extrémité.

Utiliser la tunnellisation conjointement avec le mapping d'objets natifs est également utile. Cela permet à des fonctions héritées communes d'utiliser le mapping d'objets natifs, alors que les fonctions rarement utilisées peuvent être tunnellisées. Cela peut conduire à un moindre effort total en conversion de protocole.

Annexe O
(informative)

Conversion de protocoles générique

O.1 Vue d'ensemble

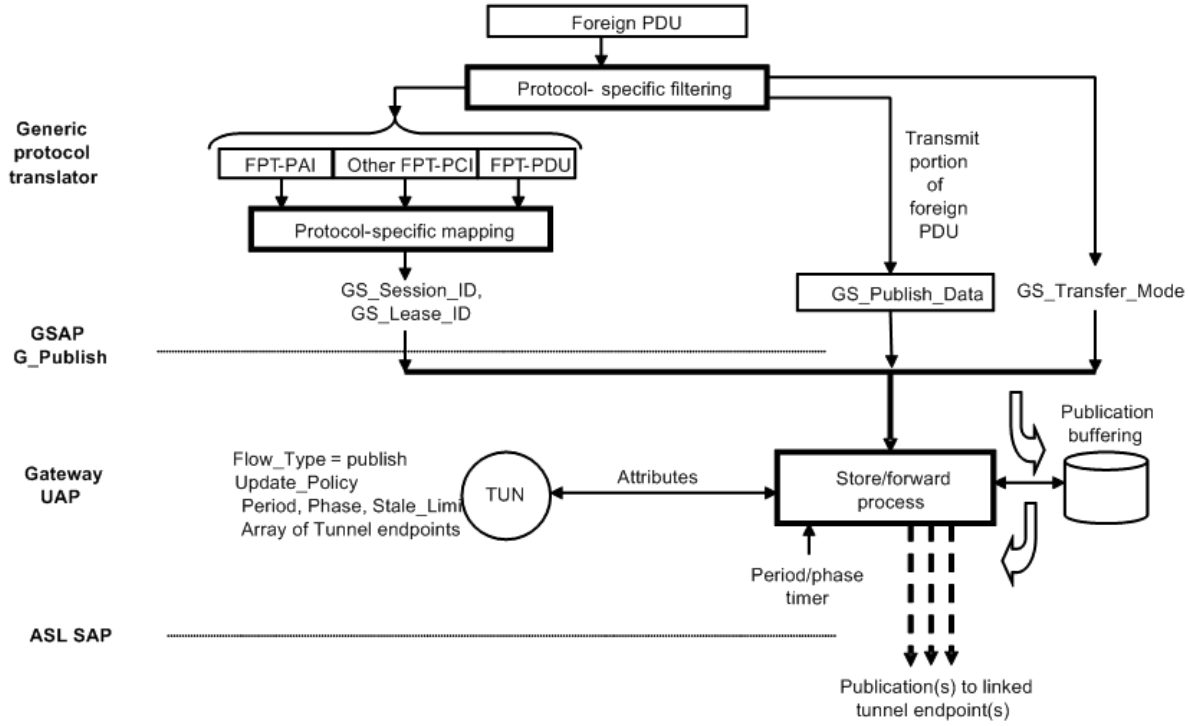
La présente norme n'inclut pas de convertisseurs de protocoles. Elle inclut des appareils pour prendre en charge la construction des convertisseurs de protocole (généralement situés au sein de passerelles) pour des protocoles communs de bus de terrain, où un tel convertisseur serait également sensible aux contraintes des réseaux d'automation sans fil de faible puissance. Sachant que des convertisseurs spécifiques de protocole ne sont pas définis dans la présente norme, tout appui pour la conversion de protocole est donc générique.

L'Annexe O fournit un exemple de la façon d'utiliser l'objet tunnel et un GIAP conceptuel pour prendre en charge des interactions communes de conversion de protocole. L'objet tunnel inclut des caractéristiques normatives pour prendre en charge la conversion de protocole.

Les convertisseurs spécifiques de protocole (pour des bus de terrain spécifiques) pourraient inclure l'Annexe O, potentiellement sous une forme modifiée. Ils pourraient également utiliser une approche différente. De tels choix ne sont pas spécifiés par la présente norme.

O.2 Publish

Une partie d'une passerelle générique est décrite à la Figure O.1, qui se rapporte à l'utilisation de la publication. Un convertisseur générique de protocole interagit avec une passerelle UAP par l'intermédiaire du GIAP. La passerelle UAP utilise l'objet TUN pour interagir avec des homologues distants par l'intermédiaire de la suite inférieure de protocole à travers le SAP de l'ASL.



Légende

Anglais	Français
Foreign PDU	PDU étrangère
Protocol-specific filtering	Filtrage spécifique à un protocole
Generic protocol translator	Convertisseur de protocole générique
Other FPT-PCI	Autre FPT-PCI
Transmit portion of foreign PDU	Partie émission d'une PDU étrangère
Protocol-specific mapping	Mapping spécifique à un protocole
Gateway UAP	UAP de passerelle
Attributes	Attributs
Store/forward process	Processus de mise en mémoire/de transmission ultérieure
Publication buffering	Placement en tampon de publication
Period/phase timer	Temporisateur période/phase
Publication(s) to linked tunnel endpoint(s)	Publication(s) au(x) point(s) d'extrémité de tunnel relié(s)
ASL SAP	SAP de l'ASL

Figure O.1 – Diagramme d'édition de conversion de protocoles générique

Une PDU étrangère est reçue par le convertisseur de protocole, et le filtrage spécifique à un protocole est appliqué. Selon le protocole, une combinaison de FPT-PAI, de tout autre FPT-PCI, et de FPT-PDU peut être nécessaire afin de déterminer les GS_Session_ID et GS_Lease_ID appropriés pour l'utilisation du GIAP dans un processus de liaison à un abonné. Le filtrage spécifique à un protocole détermine la partie de la PDU étrangère qui a besoin d'être émise (GS_Publish_Data) et des paramètres de transport spécifiques à un protocole étranger tels que la priorité (GS_Transfer_Mode). Les paramètres sont ensuite utilisés pour invoquer des services de GIAP.

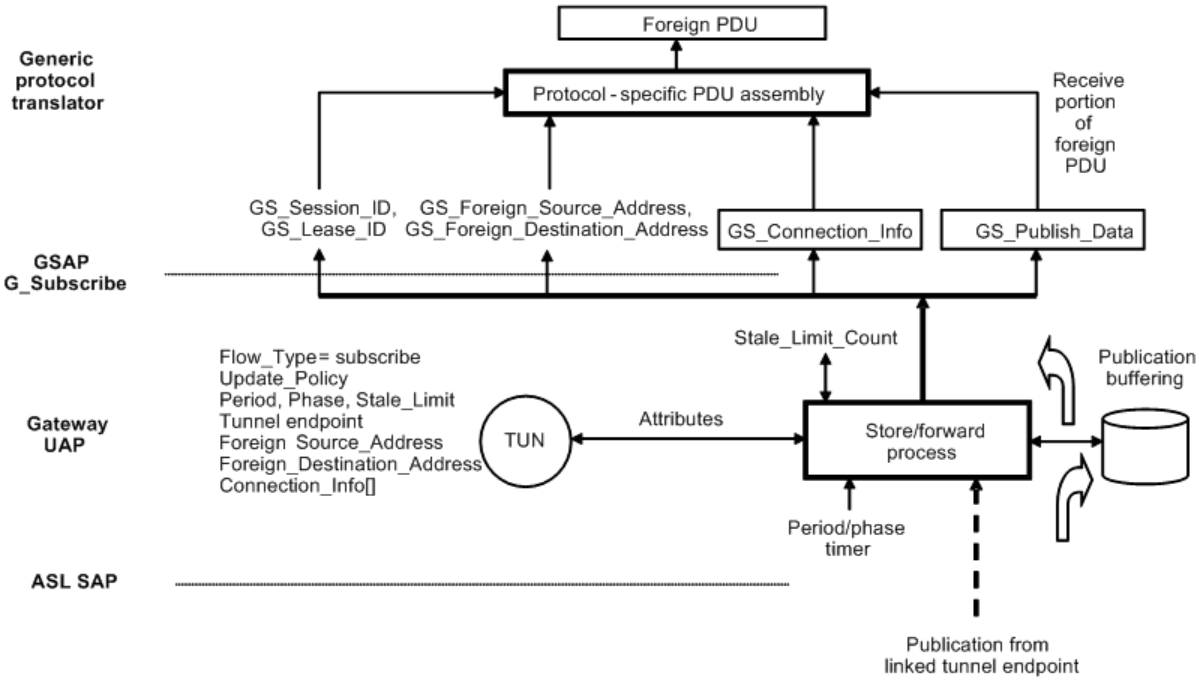
Le GS_Session_ID et le GS_Lease_ID sont utilisés par la passerelle UAP pour identifier l'objet TUN et pour récupérer les paramètres nécessaires pour des décisions de traitement en mode différé (mise en mémoire et retransmission ultérieure). GS_Publish_Data est tamponné et transmis au moment opportun selon l'Update_Policy, la période, la phase, le Stale_Limit, et

le contenu de données antérieur et actuel. Les décisions de mise en mémoire et de transmission ultérieure sont également commandées par des événements de temporisateur basés sur la période et la phase. Le SAP de l'ASL est utilisé pour transmettre les éventuels messages.

Une publication peut être envoyée à un ou plusieurs points d'extrémité selon le nombre d'éléments contenus par la matrice de points d'extrémité de tunnel.

O.3 Subscribe

Une partie d'une passerelle générique est décrite à la Figure O.2, qui se rapporte à l'utilisation de l'abonnement. Un convertisseur générique de protocole interagit avec une passerelle UAP par l'intermédiaire du GIAP. La passerelle UAP utilise l'objet TUN pour interagir avec des homologues distants par l'intermédiaire de la suite inférieure de protocole à travers le SAP de l'ASL.



Légende

Anglais	Français
Foreign PDU	PDU étrangère
Generic protocol translator	Convertisseur de protocole générique
Protocol-specific PDU assembly	Assemblage de PDU spécifiques à un protocole
Receive portion of foreign PDU	Partie réception d'une PDU étrangère
Gateway UAP	UAP de passerelle
Attributes	Attributs
Store/forward process	Processus de mise en mémoire/de transmission ultérieure
Publication buffering	Placement en tampon de publication
Period/phase timer	Temporisateur période/phase
Publication from linked tunnel endpoint	Publication à partir d'un point d'extrémité de tunnel relié
ASL SAP	SAP de l'ASL

Figure O.2 – Diagramme d'abonnement de conversion de protocoles générique

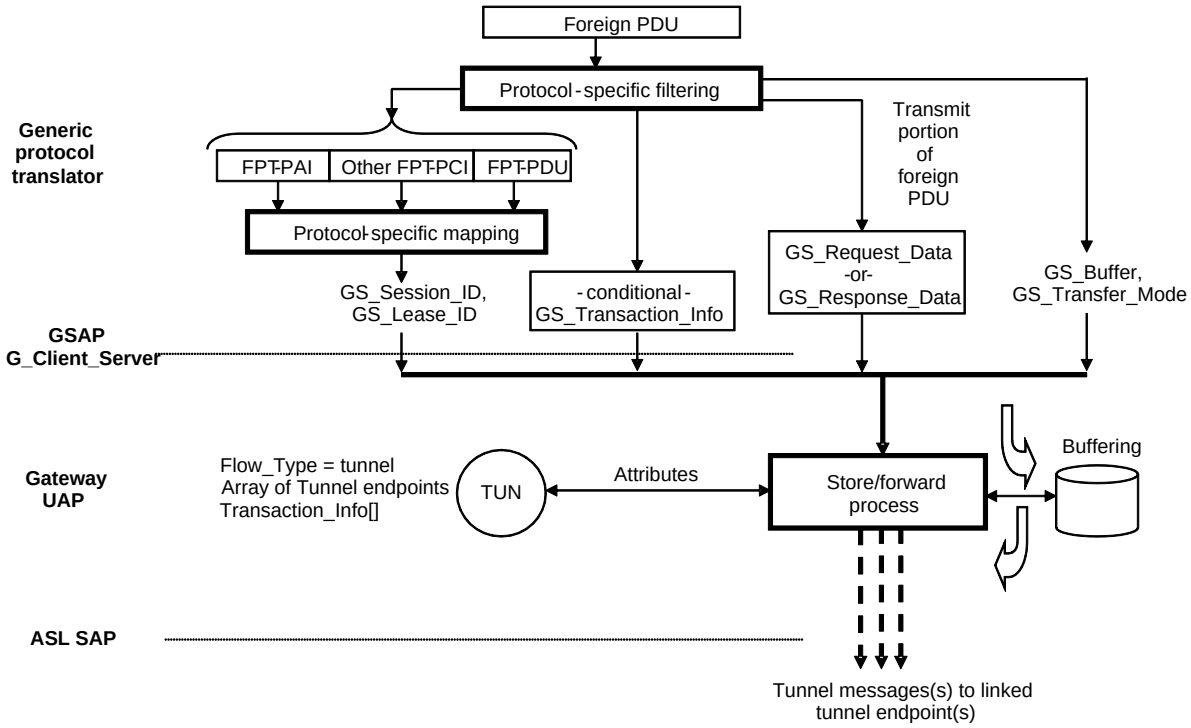
Une APDU de publication arrive à la passerelle UAP par le SAP de l'ASL. L'adressage indique un objet TUN local qui est relié à l'objet TUN d'éditeur distant. Les attributs nécessaires sont récupérés à partir de l'objet TUN pour des décisions de mise en mémoire et transmission ultérieure.

Les données de publication comportent le GS_Publish_Data issu de l'éditeur. Le placement en tampon de la publication est basé sur Update_Policy, la Period, la Phase, le Stale_Limit, et les événements de temporisateurs basés sur la Period/Phase. La transmission se produit vers le convertisseur de protocole en passant par le GIAP en fonction de l'interaction interrogée et événementielle avec le convertisseur de protocole. La passerelle UAP également met en mémoire et inclut le GS_Session_ID et le GS_Lease_ID pour que le convertisseur de protocole identifie la publication. Des informations spécifiques à la publication peuvent être mises en mémoire localement et utilisées pour réduire l'émission inutile des informations. Ces informations comprennent des informations d'adressage (GS_Foreign_Source_Address et GS_Foreign_Destination_Address) et des informations spécifiques à une connexion (GS_Connection_Info).

Le convertisseur de protocole accomplit un assemblage spécifique à un protocole pour générer la PDU étrangère.

O.4 Client

Une partie d'une passerelle générique est décrite à la Figure O.3, qui se rapporte à la transmission des messages client/serveur tunnellisés. Un convertisseur générique de protocole interagit avec une passerelle UAP par l'intermédiaire du GIAP. La passerelle UAP utilise l'objet TUN pour interagir avec des homologues distants par l'intermédiaire de la suite inférieure de protocole à travers le SAP de l'ASL.



Légende

Anglais	Français
Foreign PDU	PDU étrangère
Protocol-specific filtering	Filtrage spécifique à un protocole
Generic protocol translator	Convertisseur de protocole générique

Anglais	Français
Other FTP-PCI	Autre FTP-PCI
Transmit portion of foreign PDU	Partie émission d'une PDU étrangère
Protocol-specific mapping	Mapping spécifique à un protocole
– conditional –	– sous condition –
- or -	- ou -
Gateway UAP	UAP de passerelle
TUN	TUN
Attributes	Attributs
Store/forward process	Processus de mise en mémoire/de transmission ultérieure
Buffering	Placement en tampon
Tunnel message(s) to linked tunnel endpoint(s)	Message(s) tunnel au(x) point(s) d'extrémité de tunnel relié(s)
ASL SAP	SAP de l'ASL

Figure O.3 – Diagramme d'émission client/serveur de conversion de protocoles générique

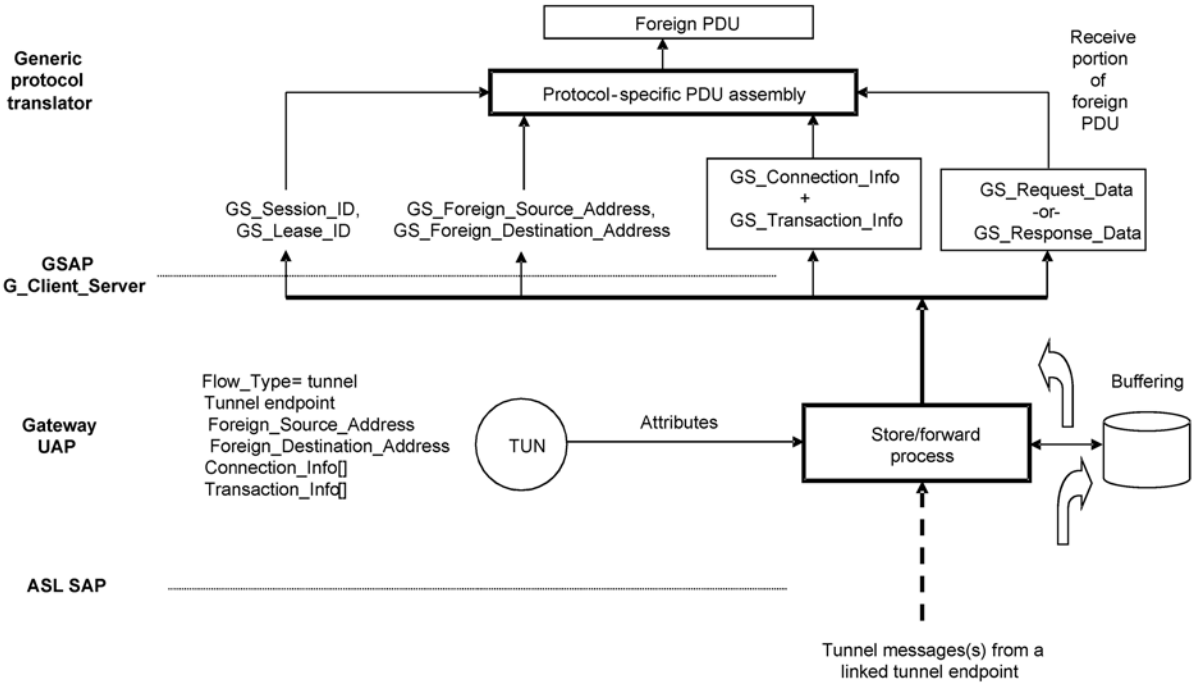
Une PDU étrangère est reçue par le convertisseur de protocole, et le filtrage spécifique à un protocole est appliqué. Selon le protocole, une combinaison de FPT-PAI, de tout autre FPT-PCI, et de FPT-PDU peut être nécessaire afin de déterminer les GS_Session_ID et GS_Lease_ID appropriés pour l'utilisation du GIAP. Le filtrage spécifique à un protocole détermine la partie de la PDU étrangère qui a besoin d'être émise (GS_Request_Data ou GS_Response_Data) et des paramètres de transport appropriés tels que la priorité (GS_Transfer_Mode). Pour des demandes, la SDU peut également spécifier GS_Transaction_Info qui doit être retourné au GIAP lorsqu'une réponse concordante arrive. Les paramètres sont ensuite utilisés pour invoquer des services de GIAP.

Le GS_Session_ID et le GS_Lease_ID sont utilisés par la passerelle UAP pour identifier l'objet TUN et pour récupérer les paramètres nécessaires pour des décisions de traitement en mode différé (mise en mémoire et retransmission ultérieure). Les informations de GIAP (GS_Request_Data ou GS_Response_Data) peuvent être tamponnées avant transmission, selon que le placement en tampon est demandé (GS_Buffer) ou non, selon le contenu antérieur du tampon, et selon qu'il est spécifié une demande ou une réponse. Le SAP de l'ASL est utilisé pour transmettre les éventuels messages.

Un message de demande de tunnel peut être envoyé à un ou plusieurs points d'extrémité selon le nombre d'éléments contenus par la matrice de points d'extrémité de tunnel. Un message de réponse de tunnel peut être envoyé à un seul point d'extrémités, mais plusieurs réponses peuvent être envoyées au même point d'extrémité au fil du temps.

O.5 Server

Une partie d'une passerelle générique est décrite à la Figure O.4, qui se rapporte à la réception des messages client/serveur tunnellisés. Un convertisseur générique de protocole interagit avec une passerelle UAP par l'intermédiaire du GIAP. La passerelle UAP utilise l'objet TUN pour interagir avec des homologues distants par l'intermédiaire de la suite inférieure de protocole à travers le SAP de l'ASL.



Légende

Anglais	Français
Foreign PDU	PDU étrangère
Generic protocol translator	Convertisseur de protocole générique
Protocol-specific PDU assembly	Assemblage de PDU spécifiques à un protocole
Receive portion of foreign PDU	Partie réception d'une PDU étrangère
Gateway UAP	UAP de passerelle
Attributes	Attributs
Store/forwards process	Processus de mise en mémoire et de transmission ultérieure
Buffering	Placement en tampon
Tunnel message(s) from a linked tunnel endpoint	Message(s) tunnel à partir d'un point d'extrémité de tunnel relié
ASL SAP	SAP de l'ASL

Figure O.4 – Diagramme de réception client/serveur de conversion de protocoles générique

Une APDU de demande ou de réponse de tunnel arrive à la passerelle UAP par le SAP de l'ASL. L'adressage indique un objet TUN local qui est relié à un objet TUN distant. Les attributs nécessaires sont récupérés à partir du TUN pour des décisions de mise en mémoire et transmission ultérieure.

Les données d'APDU de tunnel incluent soit GS_Request_Data, soit GS_Response_Data. Selon le mode tunnel, les données de réponse peuvent être tamponnées pour répondre à des demandes ultérieures issues de tampons locaux. La transmission se produit vers le convertisseur de protocole en passant par le GIAP en fonction de l'interaction interrogée et événementielle avec le convertisseur de protocole. La passerelle UAP également met en mémoire et inclut le GS_Session_ID et le GS_Lease_ID pour que le convertisseur de protocole identifie les données de tunnel. Des informations spécifiques à un message tunnel peuvent être mises en mémoire et utilisées pour réduire l'émission dupliquée inutile des informations. Cela inclut des informations d'adressage (GS_Foreign_Source_Address et GS_Foreign_Destination_Address), des informations spécifiques à une connexion

(GS_Connection_Info) et des informations spécifiques à une transaction (GS_Transaction_Info) devant être transportées dans les réponses.

Le convertisseur de protocole accomplit un assemblage spécifique à un protocole pour générer la PDU étrangère.

Annexe P
(informative)

Adaptations exemplaires du GIAP pour la présente norme

P.1 Généralités

La présente norme ne définit pas la fonctionnalité pour une passerelle complète. Elle inclut des exemples d'appui qui permettent la construction de passerelle par l'addition d'un convertisseur de protocole, et une interface matérielle et une pile pour un réseau étranger. L'Annexe P ne définit pas un convertisseur de protocole; cela pourrait être un sujet de future normalisation.

L'Annexe P fournit un exemple d'une interface conceptuelle qui serait interne à une passerelle – le GIAP, qui est censé être une abstraction du système sans fil sous-jacent. En particulier, il est censé fournir une abstraction pour le système sans fil décrit dans la présente norme.

L'Annexe P décrit une manière de mettre en œuvre le GIAP informatif en utilisant les objets et les services normatifs de la présente norme. Il ne s'agit pas d'une conception complète, mais c'est une référence pour aider à la compréhension.

Les passerelles spécifiques (pour les bus de terrain spécifiques) pourraient inclure l'Annexe P, la rendant donc normative. Elles pourraient également déterminer une approche différente qui était conforme.

Dans cette passerelle exemplaire, les services de GIAP sont mis en œuvre sous la forme d'un UAP spécialisé qui utilise des objets natifs tels que définis dans la présente norme.

P.2 Paramètres

GS_Network_Address est l'adresse IPv6.

P.3 Session

Le service de session du GIAP suit les ressources à la trace et libère les ressources lorsque la session se ferme ou expire. Les ressources comprennent des contrats de communication, transferts en masse en cours, des informations tamponnées, des ressources publication/édition/C/S dans les objets, et des abonnements d'alertes.

P.4 En leasing

Le service de location de GIAP permet l'allocation de ressources et la libération individuelle lorsque la location est fermée ou expire. Les ressources comprennent des contrats de communication et des ressources d'objets pour: le transfert en masse, C/S, P/S et les alertes.

Une location diffère d'un contrat de communication parce qu'une location alloue à la fois des ressources au sein d'une entité de passerelle et, lorsqu'elles sont nécessaires, les ressources correspondant à un contrat de communication connexe.

La spécification de plusieurs adresses IPv6 au sein de GS_Network_Address_List représente un groupe de multidiffusion. Le fait de spécifier plusieurs adresses conduira à une multidiffusion simulée par l'intermédiaire de plusieurs opérations en monodiffusion. Même s'il

s'agit d'une seule location, la multidiffusion simulée exige l'allocation des plusieurs contrats point à point et la gestion simultanée de ce contrat établi au sein de la passerelle.

GS_Resource spécifie l'élément transfert en masse pour une location (Destination_Port et OID). GS_Resource est également utilisé dans le processus de liaison de jeux concordants d'objets TNU et d'objets CON et DIS concordants. Un éditeur et un/des abonné(s) concordants spécifient des valeurs connexes dans la création de locations. Ces valeurs, avec le GS_Network_Address_List, permettent à la Array of Tunnel endpoint d'être remplie d'objets TUN liés et aux objets CON et DIS d'être alloués et liés.

Les locations d'abonné spécifient GS_Update_Policy, GS_Period, GS_Phase, et GS_Stale_Limit.

P.5 Rapport de liste d'appareils

Il n'y a aucune information d'adaptation spécifique pour cet élément.

P.6 Rapport de topologie

Il n'y a aucune information d'adaptation spécifique pour cet élément.

P.7 Rapport de programmation

Il n'y a aucune information d'adaptation spécifique pour cet élément.

P.8 Rapport de santé d'appareil

Il n'y a aucune information d'adaptation spécifique pour cet élément.

P.9 Rapport de santé de voisin

GS_Signal_Strength est en correspondance avec ED et GS_Signal_Quality est en correspondance avec LQI (voir définition en 9.1.15.2).

P.10 Rapport de santé de réseau

Il n'y a aucune information d'adaptation spécifique pour cet élément.

P.11 Durée

Il n'y a aucune information d'adaptation spécifique pour cet élément.

P.12 Client/server

P.12.1 Généralités

Le service C/S de GIAP utilise l'objet TUN ou l'IFO, selon l'établissement de la location.

P.12.2 Accès natif

Lorsque l'établissement de la location spécifie GS_Protocol_Type = 0, le protocole natif est configuré par l'intermédiaire d'un IFO, le GS_Network_Address_List est vide, et les

GS_Lease_Parameters spécifient seulement GS_Transfer_Mode, qui spécifie à son tour à la fois la priorité (priority) et l'éligibilité au rejet (discard eligibility), telles que définie à l'Article 12 pour les services read, write et execute.

Les charges utiles (GS_Request_Data et GS_Response_Data) se conforment aux formats d'APDU natifs et utilisent seulement les types de service de l'ASL: lire, écrire et exécuter. Les objets IFO dans des passerelles transfèrent ces charges utiles par l'intermédiaire des services lecture, écriture, et exécuter. GS_Transfer_Mode est utilisé avec chaque transfert pour indiquer la qualité de service, y compris la priorité, associée au transfert.

GS_Buffer est utilisé pour demander le comportement tamponné et non tamponné selon ce qui est approprié aux classifications d'attributs et de services de l'ASL.

GS_Transaction_Info est vide.

Le service C/S natif est utilisé pour adresser des objets natifs dans la passerelle.

P.12.3 Accès étranger

Lorsque l'établissement de location spécifie un GS_Protocol_Type, un protocole étranger est configuré par l'intermédiaire d'un objet TUN. GS_Network_Address_List est fourni pour établir les points d'extrémité TN distants. GS_Resource est utilisé pour déterminer si les services tunnel à deux parties ou à quatre parties s'appliquent et pour faire la concordance avec les points d'extrémité TUN au sein des appareils. Une location de client ou de serveur seule établit un tunnel à deux parties. Une paire de locations de client et de serveur avec le même GS_Resource établit un tunnel à quatre parties. GS_Lease_Parameters fournissent GS_Connection_Info aux services de Serveur comme approprié pour le protocole étranger et le GS_Transfer_Mode afin d'établir les valeurs par défaut de la qualité de service et de la priorité de transfert.

Les charges utiles (GS_Request_Data et GS_Response_Data) se conforment aux formats d'APDU étrangers, y compris la spécification de types de service étrangers et de champs spécifiques à un service étrangers. Les objets TUN dans des passerelles transfèrent ces charges utiles en utilisant les services de tunnel à deux parties et à quatre parties. GS_Cache est utilisé pour demander un comportement tamponné et non tamponné comme approprié à la configuration d'objet TUN et aux exigences de protocoles étrangers. GS_Transfer_Mode est utilisé avec chaque transfert pour indiquer la qualité de service, y compris la priorité, associée au transfert.

Le GS_Transfer_Mode spécifie la priorité et l'éligibilité au rejet, telles que définies à l'Article 12 pour le service tunnel.

GS_Transaction_Info est fourni sur des services client et retourné sur des services serveur comme approprié pour le protocole étranger.

P.13 Publish/Subscribe (éditer/s'abonner)

P.13.1 Généralités

Le service P/S de GIAP utilise l'objet TUN ou les objets CON et DIS selon l'établissement de la location.

P.13.2 Accès natif

Lorsque l'établissement de la location spécifie GS_Protocol_Type = 0, le protocole d'application natif sera édité par l'objet CON et souscrit par l'intermédiaire de l'objet DIS. GS_Network_Address_List est vide. GS_Lease_Parameters contiennent seulement

GS_Transfer_Mode afin d'établir les valeurs par défaut de la qualité de service et de la priorité de transfert.

GS_Network_Address_List est utilisé pour établir les points d'extrémité publish et subscribe. GS_Network_Address détermine l'adresse d'appareil distant. GS_Resource est utilisé pour déterminer l'objet DIS au sein cet appareil. Un objet CON local est sélectionné pour être relié à l'objet DIS distant. GS_Lease_Parameters fournissent GS_Update_Policy, GS_Period, GS_Phase, et GS_Stale_Limit pour établir le comportement périodique ou de changements d'état pour les objets CON et DIS. GS_Connection_Info est vide.

La charge utile de publication (GS_Publish_Data) est envoyée et reçue dans le format NativeIndividualValue ou NativeValueList. Les objets CON et DIS dans des passerelles transfèrent cette charge utile en utilisant le service Publish. GS_Transfer_Mode est fourni avec chaque transfert afin d'indiquer la qualité de service, y compris la priorité, associée au transfert.

Le GS_Transfer_Mode spécifie la priorité et l'éligibilité au rejet, telles que définies à l'Article 12 pour le service publish.

P.13.3 Accès étranger

Lorsque l'établissement de la location spécifie GS_Protocol_Type non égal à 0, GS_Protocol_Type est utilisé pour spécifier le protocole d'application étranger qui sera édité par l'intermédiaire des objets TUN. GS_Network_Address_List est utilisé pour établir les points d'extrémité TN distants. GS_Resource est utilisé pour faire la concordance avec les points d'extrémité TUN au sein des appareils. GS_Lease_Parameters fournissent GS_Update_Policy, GS_Period, GS_Phase, et GS_Stale_Limit pour établir le comportement périodique ou de changements d'état pour les services Publish et Subscribe. GS_Lease_Parameters fournissent GS_Connection_Info aux services Subscribe comme approprié pour le protocole étranger et le GS_Transfer_Mode afin d'établir les valeurs par défaut de la qualité de service et de la priorité de transfert.

La charge utile de publication (GS_Publish_Data) est envoyée et reçue dans le format non natif. Les objets TUN dans des passerelles transfèrent ces charges utiles en utilisant le service Publish. GS_Transfer_Mode est fourni avec chaque transfert afin d'indiquer le service, y compris la priorité, associé au transfert.

Le GS_Transfer_Mode spécifie la priorité (priority) et l'éligibilité au rejet (discard eligibility), telles que définies à l'Article 12 pour le service tunnel.

P.14 Transfert en masse

Le service de transfert en masse de GIAP est mis en œuvre par l'intermédiaire du protocole de transfert en masse et des objets IFO et UDO.

Le transfert en masse est utilisé pour le téléchargement montant/téléchargement descendant dans le mode semi-duplex. Un IFO agit comme un client. Les UDO agissent comme des serveurs. L'identificateur d'objet UDO représente la ressource cible pour l'opération. Une série de transferts par blocs d'AL est commandée par les objets d'extrémité s pour fournir la livraison ordonnée et exempte d'erreur de blocs complets d'une taille négociée. Il n'y a aucun appui dans un transfert fiable dans des couches inférieures. Un protocole de transfert à plusieurs phases (ouvrir, transférer et fermer) est utilisé. Une série de demandes et de réponses distinctes suit la taille totale du transfert. Des attributs de temporisation sont définis pour que l'UDO aide le client à déterminer les politiques de temporisation et de répétitions de tentative et éviter les erreurs d'encombrement. Une opération de chargement ou de téléchargement peut être fermée en raison d'erreurs sur l'une ou l'autre extrémité.

L'établissement de location pour des transferts en masse établit les ressources de communication nécessaires par l'intermédiaire d'un contrat de communication avant le transfert en masse.

La primitive de demande de G_Bulk_Open sert à initier un transfert en masse. L'appareil cible pour un transfert en masse est adressé par le GS_Network_Address, qui est une adresse IPv6. L'élément cible pour un transfert en masse est identifié par GS_Resource, qui contient le Transport_Port et l'OID pointant vers un UDO spécifique.

P.15 Alerte

Le service alerte de GIAP est mis en œuvre par des services d'alerte (alarmes et événements).

L'établissement de location pour des alertes établit les ressources de communication nécessaires par l'intermédiaire d'un contrat de communication pour permettre la réception d'alertes. GS_Alert_Source_ID spécifie Transport_Port, OID, et le type d'alerte.

P.16 Configuration de passerelle

Il n'y a aucune information d'adaptation spécifique pour cet élément.

P.17 Configuration d'appareil

Il n'y a aucune information d'adaptation spécifique pour cet élément.

Annexe Q
(informative)

Adaptations exemplaires du GIAP pour l'IEC 62591

NOTE Les informations suivantes ont été dérivées par analyse de l'IEC 62591 et peuvent contenir des erreurs. Voir la norme IEC réelle pour une pleine et correcte appréhension.

Q.1 Généralités

Q.1.1 Vue d'ensemble

La présente norme ne définit pas la fonctionnalité pour une passerelle complète. Elle inclut des exemples d'appui qui permettent la construction de passerelle par l'addition d'un convertisseur de protocole, et une interface matérielle et une pile pour un réseau étranger. Une telle addition exige un effort séparé pour définir le convertisseur de protocole.

L'Annexe Q décrit une interface exemplaire de passerelle, appelée le GIAP, qui est censée être une abstraction d'un système sous-jacent sans fil. En particulier, elle vise à fournir une abstraction pour le système sans fil décrit dans la présente norme, et également pour le système sans fil décrit dans l'IEC 62591.

L'Annexe Q décrit une manière de mettre en œuvre le GIAP informatif en utilisant le jeu de commande IEC 62591. Il ne s'agit pas d'une conception complète, mais c'est une référence pour aider à la compréhension.

Les passerelles spécifiques (pour les bus de terrain spécifiques) pourraient inclure l'Annexe Q, la rendant donc normative. Elles pourraient également adopter une approche différente.

Q.1.2 Référence

L'Annexe Q référence l'IEC 62591, l'IEC 61158-5-20, l'IEC 61158-6-20 et HCF_SPEC-183, qui spécifient certains des commandes et codages de champ de HART utilisés par l'IEC 62591.

Q.1.3 Adressage

Les informations d'identification et d'adressage d'appareil de l'IEC 62591 comprennent:

- Surnom: un court identificateur de 2 octets pour un appareil;
- Unique ID: un identificateur globalement unique de 8 octets formé par le OUI HCF= 0x00 1B1E + l'ID unique HART de 5 octets, conformes ensemble aux exigences relatives à EUI64Address;
- Long Tag: une chaîne lisible par l'homme de 32 octets.

L'interface GIAP utilise des adresses IPv6 logiques. La plupart des commandes de l'IEC 62591 utilisent des surnoms. Les passerelles IEC 62591 doivent mettre en œuvre la commande 841 (read network device identity using nickname, lire l'identité d'appareil réseau en utilisant le surnom) utilisant le surnom qui retourne un ID unique et une longue étiquette pour un surnom. La commande 832 (read network device identity using Unique ID, lire l'identité d'appareil réseau en utilisant le Unique ID) convertit l'ID unique en surnom et en longue étiquette d'un appareil.

Il convient de mettre en correspondance l'ID unique dans les octets de poids faible de l'adresse de GIAP plus longue.

Q.1.4 Interface de pile

L'IEC 62591 décrit son interface la plus haute comme étant une interface à la NL. La description d'interface de NL reçoit des paramètres qu'elle utilise pour invoquer une TL. Indépendamment de la description d'interface, le paquet par liaison radio encapsule l'en-tête de TL au sein d'une charge utile de la NL.

La charge utile de la TL encapsule une ou plusieurs commandes HART ou IEC 62591, pour les demandes et les réponses. L'Annexe Q décrit le mappage des services GIAP vers les commandes portées par le TL.

Q.1.5 Tunnellisation

Des passerelles IEC 62591 sont exigées pour tunnelliser des commandes HART. Cela signifie qu'une passerelle inclut un réseau étranger (l'interface d'hôte) connecté à la passerelle et la passerelle tunnellisera des commandes HART par l'intermédiaire du réseau étranger.

Q.1.6 Entités

La passerelle virtuelle, le gestionnaire de réseau, l'interface d'hôte (applications d'hôte) et l'interface réseau (appareils de réseau) sont toutes les entités de l'IEC 62591 qui mettent en œuvre (émettent et répondent à) des commandes HART et des commandes de l'IEC 62591. Le gestionnaire de réseau a la communication exclusive vers un gestionnaire de sécurité. Toute la communication entre le gestionnaire de réseau et les appareils de réseau et toute la communication entre les applications d'hôte et les appareils de réseau sont acheminées à travers la passerelle virtuelle, qui agit comme un hub de routage de commandes. La passerelle virtuelle elle-même met aussi en œuvre certaines commandes. La passerelle virtuelle communique avec les appareils de réseau par un ou plusieurs points d'accès réseau aussi bien qu'avec des appareils de réseau intermédiaires qui accomplissent le routage.

Q.1.7 Réponse différée

HART incorpore un mécanisme de réponse différée, dans lequel une première réponse spécifie que la commande a été reçue, mais que la réponse réelle est retardée en raison d'exigences de traitement étendu. Les services de GIAP exigent le traitement des réponses différées au sein de la passerelle. Une erreur est retournée si une commande qui attend un acquittement n'est pas acquittée.

Q.2 Paramètres

GS_Network_Address est une adresse IPv6 logique utilisée pour identifier un appareil spécifique de l'IEC 62591 au sein d'un réseau.

GS_Unique_Device_ID est un identificateur unique d'appareil au format EUI64Address utilisé pour identifier un appareil IEC 62591 unique. Toutes les passerelles partagent un ID unique de 0xF9 8100 0002.

GS_Network_ID indique un réseau IEC 62591 qui est accessible par la passerelle. L'IEC 62591 définit une ID de 16 bits. L'IEC 62591 spécifie une seule passerelle par réseau. Une passerelle à plusieurs modes spécifie plusieurs réseaux par passerelle et utilise l'ID de réseau pour identifier le réseau spécifique associé à une passerelle virtuelle de l'IEC 62591.

Q.3 Session

Plusieurs sessions peuvent être établies par l'intermédiaire d'une passerelle. Chaque session est utilisée pour communiquer avec un réseau spécifique tel qu'indiqué par GS_Network_ID qui est fourni lorsque la session est invoquée.

L'IEC 62591 inclut un concept différent qui est aussi appelé une session. Cette session fait référence à une session de sécurité de bout en bout. L'Annexe Q ne fait pas référence à la session de sécurité, mais à la session GIAP.

Le service de session libère des ressources de passerelle virtuelle IEC 62591 lorsqu'une session finit explicitement ou par expiration de temporisateur en utilisant les commandes suivantes:

- libérer toutes les locations;
- libérer les ressources de communication non utilisées;
- libérer le cache non utilisé.

Q.4 En leasing

Une location est utilisée pour allouer et libérer des ressources de communication spécifiques dans le contexte d'une session.

NOTE Les "services" IEC 62591 sont des ressources de chemin de communication allouées, d'un appareil demandeur (passerelle y compris) vers une destination. Des services sont demandés auprès du gestionnaire de réseau et identifiés par un ID de service. Les services ont des garanties de largeur de bande et de latence indépendantes, selon les demandes d'allocation de services. Le gestionnaire de réseau traite l'établissement et la gestion des ressources intermédiaires, telles que les chemins communs (partagés), en fonction des demandes.

Une location est établie avec la commande 799 (request service, demander un service). Cette commande est utilisée pour demander au gestionnaire de réseau une connexion à un autre appareil (un service) avec la largeur de bande et la latence spécifiées.

Le service est identifié par un ID de service (en mapping avec GS_Lease_ID).

GS_Lease_Period est établi par le convertisseur de protocoles.

GS_Lease_Type est défini par les fanions de demande de service et le domaine d'application de service.

GS_Protocol_Type est défini dans l'Annexe M.

Le surnom spécifie l'adresse de l'homologue de passerelle pour le service (en mapping avec GS_Network_Address_List qui inclut un seul GS_Network_Address). L'IEC 62591 inclut des mécanismes en monodiffusion, mais pas pour des services. Le peer-to-peer au niveau d'appareil est possible au sein du protocole, mais n'est pas recommandé en raison de soucis de sécurité.

GS_Resource n'est pas utilisé dans ce contexte et il est donc mis à 0.

La période/latence est en mapping avec GS_Lease_Parameters (GS_Period, GS_Phase, et GS_Stale_Limit).

La commande 801 (delete service, supprimer service) est utilisée pour notifier à un appareil la suppression d'un service spécifique (basé sur l'ID de service) en raison de la demande d'homologue ou d'une décision du gestionnaire de réseau.

Q.5 Rapport de liste d'appareils

Une passerelle de l'IEC 62591 est exigée pour mettre en œuvre la commande 814 (read device list entities, lire les entrées de liste d'appareils). Cette commande récupère une liste des ID uniques pour les appareils connus par la passerelle.

Tous les appareils retournés sont sur la liste d'appareils actifs. Les indications whitelist (liste blanche) et blacklist (liste noire) sont maintenues dans le gestionnaire de réseau et au sein de la passerelle.

GS_Network_Address, GS_Unique_Device_ID, GS_Manufacturer, GS_Model, et GS_Revision sont retournés pour chaque appareil.

Q.6 Rapport de topologie

Le rapport de topologie retourne une liste d'appareils (GS_Device_List), leur adresse (GS_Network_Address), et des informations connexes. Le rapport de liste d'appareil identifie les appareils dans un système.

Une passerelle de l'IEC 62591 est exigée pour mettre en œuvre la commande 834 (read network topology information, lire les informations de topologie de réseau). Cette commande est utilisée pour récupérer des informations relatives à un graphe (GS_Graph_List) pour un appareil spécifique. Les informations recherchées comprennent une liste d'ID de graphe (GS_Graph_ID) pour les graphes auxquels l'appareil participe et une liste de surnoms pour les voisins dans le graphe (associé à GS_Network_Address).

Une passerelle de l'IEC 62591 est exigée pour mettre en œuvre la commande 833 (read network device's neighbor health, lire la santé de voisin de l'appareil de réseau), qui retourne l'ensemble des voisins d'un appareil spécifique. Chaque élément dans la liste retourne le surnom du voisin (qui est mapping avec GS_Network_Address au sein de GS_Neighbor_List).

Q.7 Rapport de programmation

Le service de rapports de programmation renvoie des informations de programmation pour un appareil spécifique identifié par GS_Network_Address. Le rapport de liste d'appareils peut être utilisé pour identifier les appareils dans le système.

La commande 783 (read superframe list, lire liste de supertrames), normalement utilisée par le gestionnaire de réseau) est utilisée comme moyen de rechercher la liste de supertrames et leurs informations connexes issues d'un appareil spécifique. Les informations recherchées comprennent l'ID de supertrame (GS_Superframe_ID), le nombre d'intervalles de temps (GS_Num_Time_Slots) et des fanions en mode supertrame (HCF_SPEC-183:2013, Tableau 47).

GS_Slot_Size est fixé à 10 ms. GS_Start_Time est calculé à partir de SuperframeSlot = (Absolute Slot Number) % Superframe.NumSlots.

La commande 784 (read link list, lire la liste de liaisons, normalement utilisée par le gestionnaire de réseau) est utilisée pour récupérer des informations relatives à aux entrées de liaisons issues d'un appareil spécifique. Des entrées de liaison sont liées à l'utilisation d'intervalles de temps au sein des supertrames. Les informations recherchées comprennent le Superframe ID (GS_Superframe_ID), le nombre d'intervalles de temps dans la supertrame, la voie (GS_Channel), linkOptions (HCF_SPEC-183:2013, Tableau 46), linkType (HCF_SPEC-183:2013, Tableau 45) et le surnom (associé à GS_Network_Address) du voisin dans la liaison pour construire GS_Link_List.

GS_Channel_List contient une liste de canaux en liste blanche et en liste noire telles que définies par GS_Channel_Status pour atteindre GS_Channel_Number. GS_Channel_Number est en mapping avec Index = 0, IEEE 802.15.4 voie = 11, 2,405 MHz ... Index = 14, IEEE 802.15.4 voie = 25, 2,475 MHz. La commande 817 (read channel blacklist, lire liste noire de voies) est utilisée pour identifier le GS_Channel_Status pour chaque voie.

Q.8 Rapport de santé d'appareil

Le rapport de santé d'appareil retourne des informations relatives à la santé d'appareil pour une liste d'appareils (GS_Device_List), chacun étant identifié par GS_Network_Address.

Tous les appareils IEC 62591 mettent en œuvrent et éditent périodiquement la commande 779 (report device neighbor health, rapporter la santé de voisin de l'appareil) pour rendre les informations disponibles au gestionnaire de réseau et aux applications.

Une passerelle de l'IEC 62591 est exigée pour mettre en œuvre la commande 840 (read network device's statistics, lire les statistiques de l'appareil de réseau), qui rapporte la plupart des informations de la commande 779 (aucun statut de puissance). Cette commande utilise un ID unique pour récupérer une diversité d'informations relatives à un appareil spécifique, comprenant:

- nombre de DPDU générés par cet appareil (GS_DPDUs_Transmitted);
- nombre de DPDU arrêtés par cet appareil (GS_DPDUs_Failed_Transmission);
- nombre d'échecs de MIC de la DL (GS_DPDUs_Received, GS_DPDUs_Failed_Reception);
- ombre d'échecs de MIC de la NL (GS_DPDUs_Received, GS_DPDUs_Failed_Reception);
- nombre d'erreurs CRC (GS_DPDUs_Received, GS_DPDUs_Failed_Reception).

La commande 840 est utilisée plusieurs fois pour rassembler des informations pour chaque appareil dans la liste.

Q.9 Rapport de santé de voisin

La santé de voisin est périodiquement éditée vers le gestionnaire de réseau par la commande 780 (report neighbor health list, rapporter la liste de santé de voisin). L'intensité de signal de voisin est périodiquement éditée vers le gestionnaire de réseau par la commande 787 (report neighbor signal levels, rapporter les niveaux de signal de voisin), qui duplique les informations dans la commande 780.

G_Neighbor_Health_Report retourne une liste d'informations relatives à la qualité de la connexion au niveau de liaison pour l'ensemble de voisins d'un appareil spécifique. Le service est principalement mis en œuvre par la commande 833.

Une liste d'appareils connus par la passerelle (et chaque adresse d'appareil GS_Network_Address) peut être récupérée en utilisant le service de rapport de liste d'appareil de GIAP (G_Device_List_Report).

Une passerelle selon l'IEC 62591 est exigée pour mettre en œuvre la commande 833 (read network device's neighbor health) qui retourne une liste d'informations relatives à la qualité de connexion au niveau de la liaison pour l'ensemble de voisins d'un appareil spécifique. Chaque élément dans la liste retourne le surnom de voisin (qui est mapping à GS_Network_Address), le niveau de signal de réception en dB (GS_Signal_Strength), le nombre de paquets émis vers le voisin (GS_DPDUs_Transmitted), le nombre d'émissions échouées vers le voisin lorsqu'aucun ACK/NAK DPDU n'a été reçu (GS_DPDUs_Failed_Transmission), et les paquets reçus en provenance du voisin (GS_DPDUs_Received).

GS_Link_Status = 1 indique que le voisin est disponible pour la communication. GS_Link_Status = 0 indique que le voisin est indisponible pour la communication.

Une passerelle selon l'IEC 62591 est exigée pour mettre en œuvre la commande 840 (read network device's statistics), qui rapporte GS_DPDUs_Failed_Reception comme décrit dans l'article rapport de santé d'appareil.

GS_Signal_Quality n'est pas disponible et il est donc mis à valeur de qualité maximale.

Q.10 Rapport de santé de réseau

Le rapport de santé d'appareil et le rapport de santé de voisin sont utilisés pour déterminer GS_Device_Health_List et GS_Network_Health.

Une passerelle de l'IEC 62591 est exigée pour mettre en œuvre la commande 840 (read network device's statistics). Cette commande utilise un ID unique pour récupérer une diversité d'informations relatives à un appareil spécifique, comprenant:

- nombre de rattachements (GS_Join_Count);
- date du plus récent rattachement et heure de rattachement joints (GS_Start_Date);
- latence moyenne de la passerelle à ce nœud (GS_GPDU_Latency).

ASN est un compte de tous les intervalles de temps qui se sont produits depuis la formation du réseau. Il augmente toujours par incréments et n'est jamais réinitialisé. ASN a une longueur de cinq octets. ASN 0 correspond au moment de la naissance du réseau. GS_Start_Date et GS_Current_Date sont dérivés de l'ASN.

Q.11 Durée

Le temps de réseau selon l'IEC 62591 est mesuré par rapport au numéro d'intervalle de temps absolu 0 (ASN 0), qui est l'instant du dernier redémarrage du réseau. Ce temps progresse en incréments de 10 ms par intervalle.

La distribution de temps est configurée par le gestionnaire de réseau en utilisant la commande 971 (write neighbor property flag, écrire le fanion de propriété de voisin) pour spécifier un voisin avec les fanions de voisin (source de temps 0x01, HCF_SPEC-183:2013, Tableau 59) indiquant un voisin spécifique comme étant une source de temps. La passerelle selon l'IEC 62591 est toujours configurée comme étant la source de temps de réseau.

Le temps d'intervalle est mis à jour à travers les voisins par synchronisation par l'intermédiaire des erreurs de temps vues dans des échanges de paquets (par exemple, un champ ACK/NAK DPDU TsError).

La passerelle virtuelle doit se synchroniser avec une source extérieure de temps au moins une fois par heure. Le temps TUC est mis en correspondance avec le temps d'intervalle issu d'une référence externe à travers la passerelle. Le mapping de l'ASN 0 au TUC est diffusé à partir de la passerelle. La commande 793 (write UTC time mapping, écrire le mapping au temps TUC) est une commande de passerelle qui permet au gestionnaire de réseau d'établir le mapping du début de l'ASN 0 avec le temps TUC sur un appareil.

GS_Time est basé sur le temps TAI. Le temps TUC est basé sur le temps TAI avec des secondes intercalaires ajoutées à intervalles irréguliers. Ce service applique des mises à jour de temps à travers le GIAP. Les mises à jour de temps TAI et TUC se produisent en raison de la dérive. Le TUC ajoute les mises à jour supplémentaires en raison des secondes intercalaires. Une conversion est nécessaire du format de temps HART interne vers GS_Time et vice versa: Date HART 3 octets, heure du jour, 3 octets.

La commande 794 (read UTC time mapping, lire le mapping au temps TUC) est une commande de passerelle qui permet à un appareil ou au gestionnaire de réseau d'établir et

lire le mapping du début de l'ASN 0 avec le temps TUC. GS_Command est utilisé pour établir et lire GS_Time au sein de la passerelle pour des besoins de synchronisation. La commande 89 (set real-time clock, établir l'horloge temps réel) est utilisée pour établir le temps. La commande 90 (read real-time clock, lire l'horloge temps réel) est utilisée pour lire le temps courant.

Q.12 Client/server

Sauf spécification contraire ailleurs dans l'Annexe Q, la passerelle tunnellise toutes les commandes HART par l'intermédiaire du service C/S du GIAP. Ces commandes sont émises d'un maître vers un esclave (appareil de terrain). Le maître joue le rôle de client et l'esclave joue le rôle de serveur.

Les commandes suivent un format de demande/réponse. Des octets de données de demande sont envoyés du client vers le serveur dans GS_Request_Data. Des octets de données de réponse sont retournés du serveur vers le client dans GS_Response_Data. Les codes de réponse spécifique à une commande sont mis en mapping dans GS_Status.

Le fanion GS_Buffer est mis ou dégagé pour spécifier si, oui ou non, une commande doit être placée en tampon. Les commandes suivantes sont tamponnées:

- 0: read unique id
- 11: read unique id associated with tag
- 13: read tag, descriptor, date
- 20: read long tag
- 21: read unique id associated with long tag
- 48: read additional status
- 50: read dynamic variable assignments
- 18: write tag, descriptor, date
- 22: write long tag
- 25: write primary variable range values
- 44: write primary variable units

Plusieurs réponses de serveur peuvent être reçues avec le même GS_Transaction_ID dans le cas d'une réponse différée.

La priorité de C/S est établie par l'intermédiaire du GS_Transfer_Mode.

La priorité selon l'IEC 62591 s'inscrit dans l'un de quatre niveaux, à savoir command (la plus haute priorité), process data, normal, et alarm (la priorité la plus basse). La priorité "command" est réservée pour des paquets contenant la commande réseau, la configuration et le diagnostic. Les paquets de priorité "process data" contiennent des données de processus et sont refusés lorsque les tampons de paquets d'un appareil sont remplis aux trois-quarts. Les paquets de priorité "alarm" contiennent des alarmes et des événements. Seulement un seul paquet de priorité "alarm" est placé en tampon. Les paquets de priorité "normal" sont tous les autres paquets et sont refusés lorsque la moitié des tampons de paquets d'un appareil est remplie.

GS_Transaction_Info n'est pas exigés.

Q.13 Publish/subscribe (éditer/s'abonner)

Q.13.1 Généralités

Le service P/S du GIAP est mis en œuvre à travers la publication des commandes par les appareils conformes à l'IEC 62591 utilisant le mode continu. Les adaptateurs sont capables d'éditer pour le compte de sous-appareils non natifs. L'IEC 62591 agrège de manière native des commandes éditées où le temps s'aligne et la commande 78 (read aggregated commands, lire des commandes agrégées) n'est pas exigée.

Normalement, une passerelle s'abonne à une publication d'appareil. Au sein de G_Subscribe, GS_Publish_Data retourne les données éditées.

Il est exigé qu'une location soit acquise pour l'abonnement (obtenir GS_Lease_ID). L'établissement de location alloue des ressources entre la passerelle et l'appareil en utilisant la commande 799.

L'indication de G_Publish_Watchdog est reçue si la publication n'est pas reçue par le GS_Stale_Limit.

Q.13.2 Établissement de la location

Une location d'abonnement est établie à travers le service de location. GS_Resource spécifie les informations d'abonnement (liste de numéros de commande et de variables de processus) au service de location.

La commande 108 (write publish data mode command number, écrire le numéro de commande en mode publication de données) est utilisée comme moyen de choisir la commande devant être éditée.

Si la commande 108 spécifie la commande universelle 9 (read device variables, lire les variables d'appareil) ou la commande de pratique courante commune 33 (read device variables), des variables de processus seront assignées à des intervalles de temps pour la publication. La commande 107 (write publish data device variables, écrire les variables d'appareil de données d'édition) est utilisée pour assigner les intervalles.

La commande 103 (write publish data period, écrire la période de données d'édition) choisit la période de mise à jour minimale (GS_Period, GS_Phase) et maximale (GS_Stale_Limit) pour une publication (en incréments de 1/32 ms jusqu'à 3 600 s; la valeur demandée et la valeur réelle peuvent différer).

La commande 104 (write publish data trigger, écrire le déclencheur de données d'édition) établit une condition de déclenchement (GS_Update_Policy) pour la publication (continue/à fenêtre/en montée et un niveau) conduisant à des changements dynamiques du temps de publication. La publication se produit au moins aussi souvent que lorsque la période maximale est atteinte.

La commande 109 (publish data mode control, éditer le contrôle de mode de données) active et désactive l'édition. L'appareil de source de publication contacte le gestionnaire de réseau pour demander de la largeur de bande.

Q.13.3 Placement en tampon

Les commandes suivantes sont tamponnées:

- 1: read primary variable
- 2: read current & percent
- 3: read all variables

- 9: device variables and status
- 33: read device variables
- 123: read trend
- Device-specific (spécifique à un appareil)

Q.14 Transfert en masse

Le service de transfert en masse du GIAP correspond au transfert par blocs fourni par l'AL. L'opération permet le téléchargement montant/téléchargement descendant (GS_Mode) en mode semi-duplex ou duplex intégral, et s'appuie sur la TL pour fournir une série de transferts par blocs de niveau application. Le transport effectue la segmentation et le réassemblage selon la MTU limitée dans des couches inférieures et fournit la livraison sans erreur des blocs complets (tous les morceaux sont dans l'ordre).

L'opération utilise plusieurs phases, y compris ouvrir (G_Bulk_Open), transférer (G_Bulk_Transfer), réinitialiser, et fermer (G_Bulk_Close). De nouvelles commandes ont été créées pour exécuter ces phases. Un maître ouvre une session (commande 111) avec un esclave pour initier l'opération (GS_Transfer_ID relie les phases de cette opération). Le maître propose les longueurs de bloc (GS_Block_Size), et l'esclave peut réduire la taille. Un port (un octet) identifie la ressource cible (GS_Resource) pour l'opération (firmware, paramètres, et fichier journal). Toute la taille n'est pas énoncée (GS_Item_Size = 0) et ne peut pas être connue même par l'application (telle qu'un train continu d'échantillons organisés en blocs). Il y a un compteur d'octets sélectionné par chaque extrémité pour suivre la progression. La commande 112 est organisée de façon que la demande contienne des données de téléchargement descendant (GS_Bulk_Data) et la réponse a des données de téléchargement montant (GS_Bulk_Data). La demande crée une indication dans l'esclave; la réponse contient une indication dans le maître. La session est fermée sur des erreurs. Aucune règle n'existe sur la façon de traiter le jeu de données partielles. Le mécanisme de réponse différée est mentionné dans le statut, mais n'est pas plus décrit.

Q.15 Alerte

Le service alerte du GIAP est mis en œuvre par l'intermédiaire de plusieurs mécanismes. Les changements localement placés en tampon comprennent les mises à jour en mode par salves (changements de processus), la notification d'événement (les alarmes et les événements généraux), les changements de statut d'appareil, les changements de configuration d'appareil, les changements de topologie de réseau, et les changements de programme de réseau.

La notification de changement spécifie simplement un changement, et une action supplémentaire doit récupérer les informations altérées des tampons de passerelle. L'entité de passerelle acquitte l'arrivée d'événement aux appareils. Des publications et des alertes sont stockées dans l'entité de passerelle. L'entité de passerelle acquitte l'arrivée des alertes aux appareils.

Par exemple, la passerelle utilise souvent en interne la commande HART 115..118 pour établir une notification de changement et la commande HART 119 pour indiquer que des changements se sont produits.

Les événements sont configurés avec des numéros d'événement assignés sur une base par appareil.

La commande 116 (write event notification bit mask) configure le masque d'événement qui est utilisé pour déclencher une notification d'événement pour un événement spécifique. Le masque d'événement correspond à la commande 48 (read additional device status, lire le statut d'appareil complémentaire), qui se réfère aux tables communes 14, 17, 29, 30, 27, 31, 32, 28 et à un statut spécifique à un appareil.

La commande 117 contrôle la temporisation des notifications d'événement. La notification d'événement utilise le mode par salves pour la livraison lorsqu'un événement est déclenché. Une période anti-rebond est spécifiée pour empêcher les événements qui sont trop courts de déclencher un message à salves. Un temps de répétition de tentative (période de salves souhaitées) et une durée maximale de mise à jour (période à salves maximale) établissent la temporisation de transfert à salves si un événement déclenche un message.

La commande 118 (event notification control, contrôle de notification d'événement) est utilisée pour activer ou désactiver une notification d'événement pour un événement spécifique.

La commande 119 (acknowledge event notification, acquitter la notification d'événement) est utilisée pour acquitter la notification d'événement et dégage l'événement pour l'empêcher d'être envoyé dans des les mises à jour par salves. D'autres événements peuvent être dans la file d'attente.

La commande 115 (read event notification summary, lire le résumé de notification d'événement) est utilisée pour déterminer la configuration d'un événement en fonction d'un numéro d'événement spécifique.

Les commandes suivantes sont tamponnées:

- 119 read event notification status (marqueur temporel + statut d'appareil + commande 48).
- La commande 788 (alarm path down, alarme chemin défaillant), la commande 789 (alarm source route failed, alarme chemin de source défaillant), la commande 790 (alarm graph route failed, alarme chemin par graphe défaillant), et la commande 791 (alarm TL failed) rapportent des défaillances de communication au gestionnaire de réseau.

La commande 836 (write update notification bit mask for a device, écrire le masque de bits de la notification de mise à jour) de passerelle conforme à l'IEC 62591 enregistre un client pour des mises à jour de notifications. L'appareil est adressé par l'ID unique et reçoit un jeu de fanions de notification de changement. Des codes existent pour BurstMode, EventNotification, DeviceStatus, DeviceConfiguration, NetworkTopology (passerelle ou NM), et NetworkSchedule (passerelle ou NM). Cela est utilisé dans G_Alert_Subscription pour s'abonner (en fournissant une GS_Subscription_List avec GS_Alert_Source, GS_Subscribe, et GS_Enable pour un appareil spécifique GS_Network_Address et une catégorie spécifique GS_Category).

La commande 838 de passerelle de l'IEC 62591 (read update notification bit mask for a device, lire le masque de bits de notification de mise à jour pour un appareil) renvoie une liste des notifications de mise à jour pour un appareil. Cela est utilisé dans G_Alert_Subscription pour identifier les abonnements.

La commande 839 (change notification, notification de changement) de passerelle selon l'IEC 62591 est envoyée par la passerelle à un client et renvoie une liste comportant au maximum 10 notifications de changements (commandes placées en cache) pour un appareil. Chaque changement se traduit par une seule G_Alert_Notification.

Q.16 Configuration de passerelle

Il n'y a aucune information d'adaptation spécifique pour cet élément.

Q.17 Configuration d'appareil

Il n'y a aucune information d'adaptation spécifique pour cet élément.

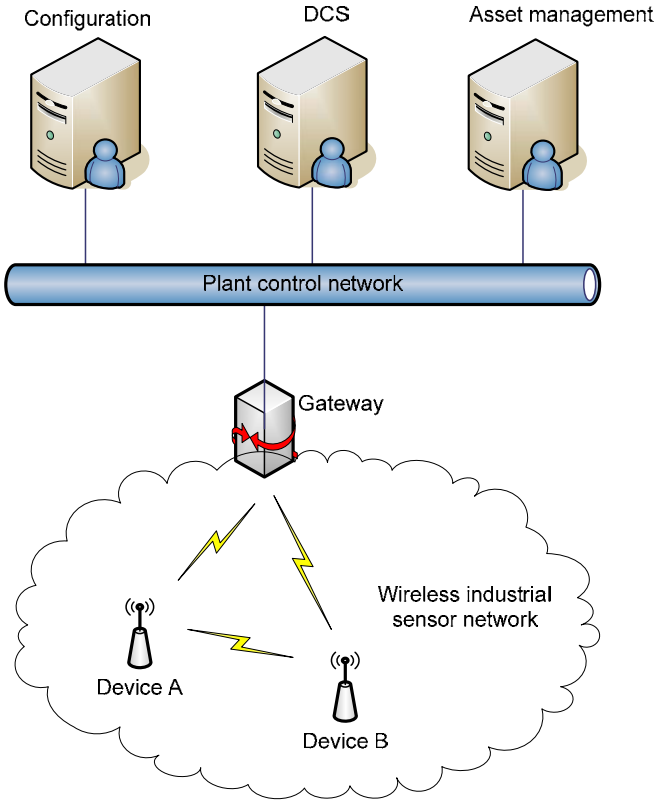
Annexe R
(informative)

Interface système hôte aux appareils conformes à la norme via une passerelle

R.1 Contexte

R.1.1 Modèle de référence d'intégration de système hôte

Un modèle simplifié de référence pour l'intégration de système hôte/appareil conforme à une norme est décrit à la Figure R.1.



Légende

Anglais	Français
Configuration	Configuration
DCS	Système de commande distribué
Asset management	Gestion d'actifs
Plant control network	Réseau de commande d'installation
Gateway	Passerelle
Device A	Appareil A
Wireless industrial sensor network	Réseau de capteurs industriels sans fil
Device B	Appareil B

Figure R.1 – Modèle de référence d'intégration de système hôte

R.1.2 Outils de gestion d'actifs

La gestion d'actifs implique de superviser la santé des biens de systèmes en surveillant des conditions relatives à la santé afin d'identifier un problème potentiel avant que le processus ou l'exploitation d'installation ne soit affecté(e). Les systèmes hôtes fournissent un outil de gestion d'actifs ou un jeu d'outils pour accomplir la fonction de gestion d'actifs, dans le but d'abaisser les coûts de service, de réduire le temps de panne, et d'assurer que les niveaux appropriés de qualité des produits sont atteints.

R.1.3 Outils de configuration

Une fois la conception du système établie et les composants du système identifiés, le fonctionnement des composants dans le système global doit être configuré. Les systèmes hôtes fournissent un outil de configuration ou un jeu d'outils qui prend en charge la configuration des composants du système et en définit le fonctionnement.

R.1.4 Système de commande distribué

Un système de commande distribué (DCS) est un système de commande qui prend en charge un processus dans lequel les éléments de commande sont géographiquement distribués. Ces éléments distribués sont connectés par des réseaux de communication, qui sont utilisés pour communiquer avec les éléments distribués.

R.1.5 Passerelle

Une passerelle connecte les systèmes hôtes au réseau. Voir l'Annexe U pour plus d'informations relatives à la passerelle.

R.2 Intégration de données d'application d'appareil avec des systèmes hôtes

R.2.1 Généralités

Il existe deux moyens génériques pour les systèmes hôtes d'intégrer des données d'application issues d'appareils connectés:

- intégration par l'intermédiaire d'un mapping de protocoles; et
- intégration par l'intermédiaire de la tunnellisation de protocoles.

R.2.2 Intégration de protocoles natifs par l'intermédiaire du mapping

Les systèmes hôtes existants peuvent intégrer des données d'application d'appareil en effectuant un mapping des relations entre les appareils et les données avec le traitement des informations accompli par le système hôte existant. Cette fonction de mapping est habituellement accomplie par une passerelle entre le système hôte existant et le réseau de capteurs industriels sans fil (WISN).

R.2.3 Intégration de protocoles d'appareil hérités par l'intermédiaire de la tunnellisation

Les systèmes hôtes peuvent intégrer des données d'application issues appareils hérités existants qui utilisent la capacité de tunnellisation d'application du WISN de la même manière par laquelle ils intègrent actuellement des données d'application issues d'appareils hérités.

R.3 Outil de configuration de système hôte

R.3.1 Généralités

Les systèmes hôtes prennent habituellement en charge l'une et/ou l'autre de deux méthodes génériques d'intégration pour configurer des appareils de terrain:

- langage de description électronique de produit (EDDL);
- outil d'appareil de terrain/gestionnaire de type d'appareil (FDT/DTM).

R.3.2 Configuration d'hôte en utilisant le langage de description d'appareil électronique

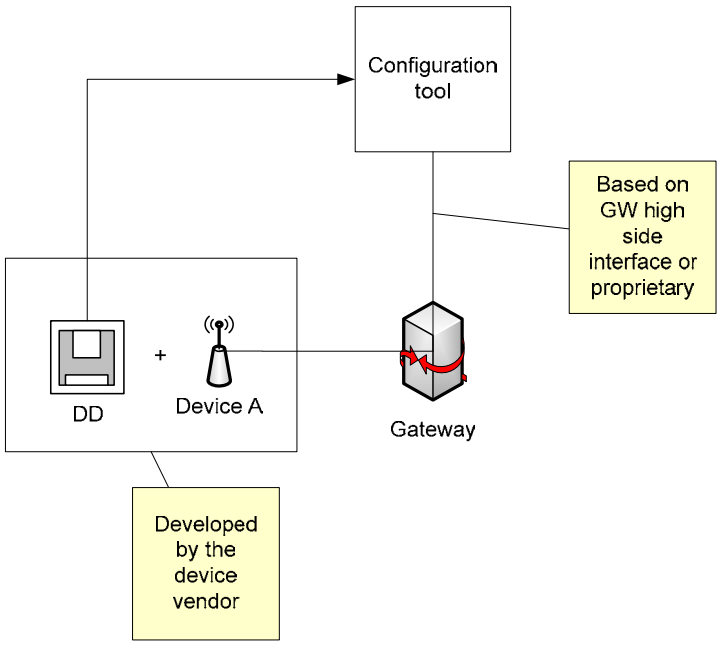
L'IEC 61804-3, EDDL, décrit un langage générique pour décrire des propriétés d'appareil d'automation. EDDL peut décrire des fonctions d'appareil, des interactions prises en charge par un appareil, des objets pris en charge par un appareil et d'autres propriétés.

EDDL est utilisé par un fournisseur d'appareil pour créer un fichier définition d'appareil électronique (EDD) qui correspond à un appareil particulier. Un fichier d'EDD est un fichier texte en ASCII structuré indépendant vis-à-vis de tout système d'exploitation et de tout système d'automation qui décrit les capacités d'un appareil pour permettre l'intégration de l'appareil avec un système DCS hôte. Cette indépendance permet à des fournisseurs de décrire leurs appareils d'une manière qui permet l'interopérabilité indépendante et l'interopérabilité contrainte vis-à-vis de tout fournisseur de l'appareil à travers des systèmes hôtes. Les fichiers EDD décrivent des données d'appareil, des caractéristiques d'interface utilisateur souhaitées par un fournisseur d'appareil, et le traitement de commandes d'appareil, tel que l'ordonnancement et la temporisation des commandes.

Les DCS d'hôte fournissent des outils pour interpréter les fichiers d'EDD afin de configurer et manipuler l'appareil, comme dans le cas de la surveillance ou celui du traitement de paramètres, pour prendre en charge des applications de commande.

Les fichiers EDD sont définis par des fournisseurs d'appareil et ils sont soumis à essai par l'organisation prenant en charge le bus de terrain approprié.

La Figure R.2 représente la configuration en utilisant un fichier DD.



Légende

Anglais	Français
Configuration tool	Outil de configuration
Based on GW high side interface or proprietary	Basée sur l'interface de côté haut de la passerelle (GW) ou propriétaire
Gateway	Passerelle
Device A	Appareil A

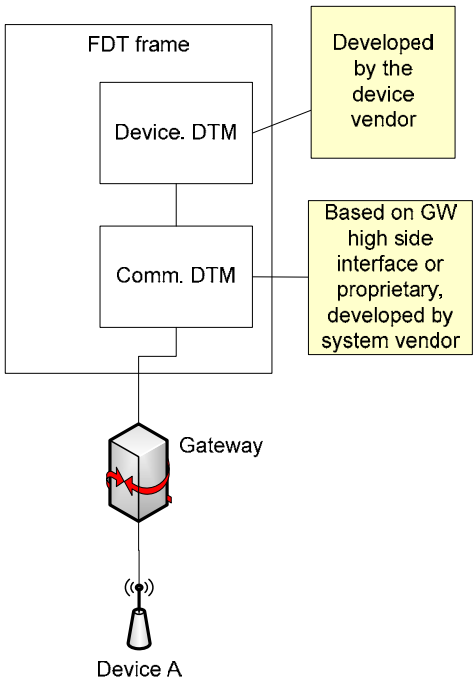
Anglais	Français
DD	DD (définition d'appareil)
Developed by the device vendor	Développée par le vendeur d'appareil

Figure R.2 – Configuration utilisant une définition d'appareil électronique

R.3.3 Configuration de l'hôte utilisant l'outil d'appareil de terrain/gestionnaire de type d'appareil

La fonctionnalité de l'appareil décrite par l'EDD est limitée par l'IEC 61804-3. La fonctionnalité supplémentaire d'appareil (le cas échéant) qui ne peut pas être décrite par l'intermédiaire de l'EDD peut être prise en charge par l'intermédiaire de prises enfichables ou à fixation directe propriétaires. Pour fournir cette plus grande prise en charge, la technologie FDT/DTM peut être utilisée. La technologie FDT/DTM requiert, par exemple, la prise en charge d'application de PDU FDT dans le DCS. Pour plus d'informations relatives à la FDT/DTM, consulter le FDT Group.

La Figure R.3 représente une configuration utilisant l'approche FDT/DTM.



Légende

Anglais	Français
FDT frame	Trame FDT
Developed by the device vendor	Développée par le vendeur d'appareil
Device. DTM	DTM d'appareil
Comm. DTM	DTM de communication
Gateway	Passerelle
Device A	Appareil A
Based on GW high side interface or proprietary, developed by system vendor	Basée sur l'interface de côté haut de la passerelle (GW) ou propriétaire, développée par le vendeur de système

Figure R.3 – Configuration utilisant l'approche FDT/DTM

R.4 Intégration appareil de terrain/systèmes de commande distribués

R.4.1 Généralités

Les systèmes de commande distribués sont habituellement composés d'appareils tels que contrôleurs, stations d'interface homme-machine (IHM), serveurs d'antécédents de données, applications avancées, etc. Les stations IHM, les serveurs d'antécédents, et les applications avancées utilisent souvent des interfaces avec une sémantique riche de données, telle que l'OPC. La communication avec des contrôleurs utilise habituellement des protocoles plus simples, tels que Modbus, ou l'Ethernet haut débit de la fondation Fieldbus (Foundation Fieldbus high speed Ethernet (FF-HSE).

R.4.2 Fondation Fieldbus – Ethernet haut débit

L'intégration de données d'application avec le FF-HSE peut, par exemple, être accomplie en mappant les données d'application natives à des blocs de transducteurs FF. Les objets d'application ont un mapping aux blocs FF, alors que les attributs d'objets sont mis directement en correspondance aux paramètres de bloc FF.

R.4.3 Modbus

Les données d'application peuvent s'intégrer avec Modbus en assignant une adresse Modbus à la passerelle. La passerelle peut alors présenter un ensemble de tables de registre aux maîtres Modbus. Chaque attribut d'objet peut être mis en correspondance avec un registre spécifique. Le système hôte peut fournir la prise en charge automatisée pour le mapping, ou le mapping peut être accompli manuellement par l'utilisateur.

R.4.4 Connectivité ouverte pour l'automation industrielle

La connectivité ouverte pour l'automation industrielle (OPC) permet à des applications client d'accéder à des données d'une façon cohérente par l'intermédiaire d'un serveur OPC en référençant les données en utilisant une construction Tag.Parameter.

Un client OPC peut être pris en charge par un serveur OPC dans le système hôte ou par une interface OPC de côté haut fournie par une passerelle à un système conforme à une norme.

Par exemple, la présente norme fournit la valeur, la qualité, et les informations de marqueur temporel dans des publications de données, à l'appui de l'accès serveur OPC à des données en ligne. Des alarmes et des événements natifs fournissent également l'appui pour la notification au client OPC.

Le client OPC peut spécifier le Tag.Parameter en utilisant le nom de l'appareil pour Tag, et un ensemble unique d'un nom et d'un attribut pour représenter le paramètre (par exemple, TI101.AITB1.PV). Dans le serveur OPC, Tag est mise en correspondance avec l'appareil, l'instance d'objet est en correspondance avec une instance particulière d'objet d'un UAP particulier, et le nom d'attribut est en correspondance avec l'identificateur d'attribut particulier de l'instance d'objet référencée.

R.5 Passerelle

R.5.1 Généralités

La configuration de système hôte des applications résidant dans la passerelle elle-même, y compris le mapping des données (s'il y a lieu), est définie par le réseau de commande d'installation, qui est l'interface de côté haut de la passerelle qui couple le WISN dans un système de commande de plus haut niveau. Cela inclut, par exemple, la configuration d'une application de gestion de système ou d'une application de tunnellisation. Par conséquent, l'Annexe R décrit dans les grandes lignes le type d'informations qui ont besoin d'être configurées pour la prise en charge de la passerelle.

R.5.2 Appareils pris en charge

Un outil de configuration de système hôte peut avoir besoin d'établir le complément d'appareils conformes à une norme avec lesquels la passerelle communiquera.

R.5.3 Abonnement à des données

Un outil de configuration de système hôte peut avoir besoin d'établir la configuration des objets dispersion dans la passerelle pour les données que la passerelle recevra par l'intermédiaire de la publication.

R.5.4 Publication de données

Un outil de configuration de système hôte peut avoir besoin d'établir la configuration des objets Concentrator dans la passerelle pour les données que la passerelle publiera elle-même.

R.5.5 Accès client/serveur

Des communications C/S non relatives à une gestion peuvent, par exemple, être établies par la passerelle sur une base selon la nécessité par l'intermédiaire d'objets d'interface.

R.5.6 Réception d'alertes

Un outil de configuration de système hôte peut avoir besoin d'établir les catégories d'alertes associées à un(des) objet(s) récepteur(s) d'alertes (ARO) résidant dans une passerelle.

R.6 Prise en charge d'application de gestion d’actifs

R.6.1 Généralités

Un outil de gestion d’actifs peut accéder à des informations relatives à un appareil qui sont soit stockées dans la passerelle, soit accessibles par l'intermédiaire de celle-ci, en utilisant des services de réseau de commande d'installation.

Une passerelle peut accéder aux informations directement à partir d'un appareil de terrain pour satisfaire aux demandes de gestion d’actifs. La passerelle peut, par exemple, utiliser des services de C/S pour lire des données, écrire des données ou exécuter une méthode particulière sur une instance d'objet particulière au sein de l'appareil sans fil.

Une passerelle peut agir comme un sas pour des information de biens directement d'un bien vers une application de gestion d’actifs par l'intermédiaire d'un tunnel de réseau de commande d'installation si le réseau de commande d'installation prend en charge une telle capacité de tunnellisation.

R.6.2 Outil d'appareil de terrain/gestionnaire de type d'appareil

Un DTM peut être fourni par un fournisseur d'appareil pour fournir des informations de processus et d'appareil à un outil de gestion d’actifs. Un système hôte prenant en charge un PDU FDT peut utiliser le DTM de l'appareil et un DTM de communication pour la passerelle pour acquérir les informations nécessaires pour gérer l'appareil par l'intermédiaire de la passerelle.

R.6.3 HART

Un appareil conforme à la norme peut être forcé à apparaître comme un appareil HART[16] natif sur une application de gestion d'actifs HART (ASM) de différentes façons:

- Manuellement ou en utilisant l'automation avec conversion soit explicitement codée, soit guidée par des données, les règles fournissent un fichier source DD de HART pour l'appareil. Le fichier DD HART peut être passé à travers un marqueur à jetons (tokenizer) HART pour produire des fichiers binaires représentant le contenu du DD. La plupart des clients de HART utilisent le format binaire des fichiers DD.
- Des commandes normalisées peuvent être définies dans HART pour intégrer l'ASM avec la présente norme, telles que les commandes HART pour READ_IEC62734_ATTRIBUTE, WRITE_IEC62734_ATTRIBUTE et EXECUTE_IEC62734_METHOD.
- Des tables de mapping dans la passerelle peuvent être utilisées pour définir le mapping de valeurs qui diffère entre la présente norme et HART, comme dans le cas des indices en unités d'étude.

R.6.4 OPC

La connectivité ouverte pour l'automation industrielle (OPC) permet à des applications client d'accéder à des données d'une façon cohérente par l'intermédiaire d'un serveur d'OPC. Un client OPC peut être pris en charge par un serveur OPC dans le système hôte ou par une interface OPC de côté haut fournie par une passerelle à un système conforme à une norme.

Par exemple, des informations de santé d'appareil peuvent être fournies par le serveur OPC à un client OPC.

---

16 HART est l'appellation commerciale de HCF. Cette information est donnée à l'intention des utilisateurs de la présente norme et ne signifie nullement que l'IEC approuve ou recommande l´emploi exclusif du (des) produit(s) ainsi désigné(s). La conformité à ce profil n'exige pas l'utilisation de la marque déposée. L'utilisation des marques déposées exige l'obtention préalable d'autorisations auprès du propriétaire des marques.

Annexe S
(informative)

Vecteurs d'essai de fonctionnement de clés symétriques

S.1 Échantillons de DPDU

S.1.1 Généralités

[INGREDIENTS]

- TsDur: 10464 [2^-20sec]
- Data DPDU Source EUI64: 0x00 00 00 00 00 00 00 01
- Data DPDU Key: 0xC0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
- Data DPDU Sequence Number: 0x04
- TAI Time[TAINetworkTimeValue]: 0x00 01 02 03 04 05
- Channel: 0x02
- Data DPDU Headers: 0x10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28
- Data DPDU Payload: 0x30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 55 56 57 58 59 5A 5B
- ACK DPDU Source EUI64: 0x00 00 00 00 00 00 00 02
- ACK DPDU Sequence Number: 0x05
- ACK DPDU Headers: 0x10 11 12 13 14 15 16 17 18

S.1.2 DPDU avec DMIC32 attendu

[PRE-PROCESSED MATERIAL]

- Data DPDU Nonce: 0x00 00 00 00 00 00 00 01 04 08 0C 10 14
- Data DPDU MIC: 0xBF 5A BB 7C
- ACK DPDU Nonce: 0x00 00 00 00 00 00 00 02 04 08 0C 10 15
- ACK DPDU authentication vector: 0x10 11 12 13 14 15 16 17 18 BF 5A BB 7C
- ACK DPDU MIC: 0x74 F0 41 B3

[DELIVERABLE]

- Data DPDU: 0x10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 55 56 57 58 59 5A 5B BF 5A BB 7C
- ACK DPDU: 0x10 11 12 13 14 15 16 17 18 74 F0 41 B3

S.1.3 DPDU avec ENC-DMIC32 attendu

[PRE-PROCESSED MATERIAL]

- Data DPDU Nonce: 0x00 00 00 00 00 00 00 01 04 08 0C 10 14
- Encrypted Data DPDU Payload: 0x23 F4 C4 3F BA 9B E4 3E D8 9B FD 36 A8 76 C7 99 27 14 E0 42 94 94 DE 64 B2 6B 14 18 51 9F 8D 11 36 F4 09 17 6B D6 A6 75 07 B1 D2 90
- Data DPDU MIC: 0xD0 F6 B2 65

- ACK DPDU Nonce: 0x00 00 00 00 00 00 00 02 04 08 0C 10 15
- ACK DPDU authentication vector: 0x10 11 12 13 14 15 16 17 18 D0 F6 B2 65
- ACK DPDU MIC: 0x26 AB 87 D2

[DELIVERABLE]

- Data DPDU: 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 23 F4 C4 3F BA 9B E4 3E D8 9B FD 36 A8 76 C7 99 27 14 E0 42 94 94 DE 64 B2 6B 14 18 51 9F 8D 11 36 F4 09 17 6B D6 A6 75 07 B1 D2 90 D0 F6 B2 65
- ACK DPDU: 0x10 11 12 13 14 15 16 17 18 26 AB 87 D2

S.2 Échantillons de TPDU

S.2.1 Généralités

[INGREDIENTS]

- TPDU time creation[TAINetworkTimeValue]: 0x00 01 02 03 04 05
- Key: 0xC0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
- Crypto Key Identifier Mode: 0x00
- Crypto Key Identifier = 0x10
- Source EUI64Address: 0x00 00 00 00 00 00 00 01
- Source IPv6Address: 0xFE 80 00 00 00 00 00 00 00 00 00 00 00 00 00 01
- Dest IPv6Address: 0xFE 80 00 00 00 00 00 00 00 00 00 00 00 00 00 02
- Source Port: 0x00 01
- Dest port: 0x00 02
- TSDU (Application Layer Payload): 0x10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B

S.2.2 TPDU avec ENC-DMIC32 attendu:

[PRE-PROCESSED MATERIAL]

- TPDU Pseudo header: 0xFE 80 00 00 00 00 00 00 00 00 00 00 00 00 00 01 FE 80 00 00 00 00 00 00 00 00 00 00 00 00 00 02 00 2B 00 11 00 01 00 02
- TPDU Nonce: 0x00 00 00 00 00 00 00 01 04 08 0C 10 FF
- TPDU Security header: 0xA0 0C 10

[DELIVERABLE]

- TPDU: 0x 00 01 00 02 00 23 00 00 A0 0C 10 8E 7C 0B B9 8B CD 15 7E 59 CE 71 18 14 B7 05 FE C2 6A F1 C3 9D 05 B9 FD E6 5F 16 C9 DE 37 DE BE

S.2.3 TPDU avec TMIC-32 attendu:

[PRE-PROCESSED MATERIAL]

- TPDU Pseudo header: 0xFE 80 00 00 00 00 00 00 00 00 00 00 00 00 00 01 FE 80 00 00 00 00 00 00 00 00 00 00 00 00 00 02 00 2B 00 11 00 01 00 02
- TPDU Nonce: 0x00 00 00 00 00 00 00 01 04 08 0C 10 FF
- TPDU Security header: 0x20 0C 10

[DELIVERABLE]

- TPDU: 0x 00 01 00 02 00 23 00 00 20 0C 10 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B 7E 8C 35 57

Annexe T
(informative)

En-têtes de liaison de données et de réseau pour des demandes de rattachement

T.1 Vue d'ensemble

L'Annexe T décrit l'en-tête de la DL et l'en-tête de la NL pour une demande de rattachement typique.

T.2 En-tête MAC (MHR)

L'en-tête MAC pour messages de rattachement est montré dans le Tableau T.1. La convention IEEE montre le bit 0 sur la droite, qui est l'ordre nominal de transmission. D'après la convention IEEE 802.15.4, les champs Sequence Number et Addressing du MHR, lorsqu'ils sont considérés comme des entiers non signés, sont d'abord transmis en mode LSB (octet de poids le plus faible).

NOTE En fait, IEEE 802.15.4 2,4 GHz DSSS transmet des quartets de quatre bits simultanément sous la forme d'un signal à étalement de spectre de 32 puces; il n'existe donc pas de "premier" ou "dernier" bit transmis dans le quartet. Néanmoins, le quartet de poids de bits inférieur dans un octet, lorsqu'il est interprété en tant que valeur Unsigned8, est transmis avant le quartet de poids de bits supérieur du même octet tandis que l'octet de poids inférieur est transmis avant l'octet de poids supérieur.

Le Tableau T.1 suit la convention de la présente norme en montrant le bit 7 sur la gauche.

Tableau T.1 – MHR échantillon pour demande de rattachement

<table>
<tr><th rowspan="2">Sous-trame</th><th rowspan="2">Nombre d'octets</th><th colspan="8">bits</th></tr>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td rowspan="2">Contrôle de trame</td><td rowspan="2">2</td><td>Réservé</td><td>PAN ID Compress = 1 (yes)</td><td>ACK Request = 0 (no)</td><td>Frame Pending = 0 (no)</td><td>Security Enabled = 0 (no)</td><td colspan="3">Frame Type = 1 (Data)</td></tr>
<tr><td colspan="2">Source Addressing Mode= 3 (64-bit)</td><td colspan="2">Frame Version = 1</td><td colspan="2">Dest Addressing Mode=2 (16-bit)</td><td colspan="2">Réservé</td></tr>
<tr><td>Numéro de séquence</td><td>1</td><td colspan="8">(déterminé par le DLE au moment de la transmission)</td></tr>
<tr><td rowspan="3">Adressage</td><td>2</td><td colspan="8">PAN ID (LSB), depuis l'annonce</td></tr>
<tr><td>2</td><td colspan="8">Adresse de destination (16 bits, LSB), depuis l'annonce</td></tr>
<tr><td>8</td><td colspan="8">Adresse source (64 bits, LSB), adresse EUI64 de l'appareil</td></tr>
</table>

T.3 En-tête de DL (DHR)

L'en-tête DL pour messages de rattachement est montré dans le Tableau T.2. Cet exemple suppose que l'annonce ne spécifie pas de lent saut de voie.

Tableau T.2 – DHR échantillon pour demande de rattachement

Sous-titre	octets	bits							
		7	6	5	4	3	2	1	0
DHDR	1	ACK/NAK DPDU needed = 1 (yes)	Signal quality in ACK/NAK DPDU = 0 (no)	Request EUI64Address = 0 (no)	Include DAUX = 0 (no)	Include slow channel hopping-offset = 0 (no)	Clock recipient = 1(yes)	DL version = 00	
DMXHR	1	Reserved=0			Key identifier mode = 1		Security level = 1 (MIC-32)		
	1	Crypto key identifier = 0: K_global							
DAUX	0	(absent par réglage DHDR)							
DROUT	1	Compress = 1	Priority = 0 (irrelevant)				DlForwardLimit = 1		
	1	GraphID (Unsigned8) =0 (Single hop source routing)							
DADDR	1	DE = 0	LH = 0	ECN = 0	Réservé				
	1	SrcAddr = 0 (Use EUI64Address in MHR)							
	1	DestAddr = 0 (Use DL16Address in MHR)							

T.4 En-tête de NL

L'en-tête réseau pour messages de rattachement est montré dans le Tableau T.3.

Tableau T.3 – En-tête réseau pour messages de rattachement

octets	bits							
	7	6	5	4	3	2	1	0
1	LOWPAN_IPHC dispatch = 011			LOWPAN_IPHC encoding (bits 8..12) = 11 101				
2	LOWPAN_IPHC encoding (bits 0..7) = 0111 0111							

Annexe U
(informative)

Rôle de la passerelle

U.1 Généralités

U.1.1 Vue d'ensemble

Le but primaire d'une passerelle telle que décrite par la présente norme est de permettre à des applications de niveau hôte d'interagir avec des appareils de terrain sans fil. Une grande base d'applications installée existe, y compris des appareils d'automation, des contrôleurs, et des systèmes de surveillance, qui utilisent ensemble de nombreux protocoles hérités, ce qui exige une conversion de protocoles pour interagir avec des appareils de terrain sans fil. Une telle conversion de protocole peut être présente dans la passerelle et également dans des adaptateurs aux appareils de terrain câblés hérités. Dans la présente norme, le terme adaptateur est utilisé pour identifier des appareils qui convertissent un protocole de bus de terrain câblés en un protocole de bus de terrain sans fil pour le compte d'un ou plusieurs appareils de terrain.[17] Une telle conversion de protocole sert généralement à tunnelliser un protocole étranger à travers un réseau sans fil tel que décrit dans la présente norme ou à convertir un protocole hérité vers un format natif de la présente norme et vice versa. Le terme appareil de terrain natif se réfère à un appareil de terrain qui fonctionne exclusivement par l'intermédiaire de l'utilisation des objets natifs, des interfaces natives et du contenu de message natif tels que définis dans la présente norme.[18] Il est également possible d'écrire ou de modifier des applications de niveau hôte pour utiliser directement le protocole d'application natif, réduisant ou éliminant le besoin de conversion de protocole au sein d'une passerelle.

NOTE Les exemples fournis à l'Annexe U sont symétriques, potentiellement applicables sans modification aux passerelles et aux adaptateurs. Techniquement, une passerelle est une interface d'une voie de communication à une autre voie de communication, où la conversion de protocole est utilisée à une ou plusieurs couches de la suite de protocoles. Il existe un précédent établi dans l'industrie d'automation consistant à (incorrectement) utiliser le terme adaptateur pour identifier des appareils qui convertissent un protocole de bus de terrain câblés en un protocole de bus de terrain sans fil pour le compte d'un ou plusieurs appareils de terrain.

La description du rôle de passerelle se rapporte aux capacités suivantes:

- Interfacer des applications étrangères de niveau hôte:
  - directement aux appareils de terrain "natifs" (c'est-à-dire conformes à la présente norme); et
  - indirectement aux appareils de terrain câblés hérités via des adaptateurs hérités.
- Interfacer des applications de niveau hôte à plusieurs systèmes sans fil, y compris une combinaison d'un ou plusieurs système sans fil tels que décrits dans la présente norme et un ou plusieurs systèmes sans fil étrangers, par l'intermédiaire d'un seul appareil (conceptuel) avec une interface commune de côté haut.

---

[17] L'utilisation du terme adaptateur n'est pas uniforme. Techniquement, un adaptateur est une interface d'une CPU avec une voie de communication. Techniquement, une passerelle est une interface d'une voie de communication à une autre voie de communication, où la conversion de protocole est utilisée à une ou plusieurs couches de la suite de protocoles. Il existe un précédent établi dans l'industrie d'automation consistant à (incorrectement) utiliser le terme adaptateur pour identifier des appareils qui convertissent un protocole de bus de terrain câblés en un protocole de bus de terrain sans fil pour le compte d'un ou plusieurs appareils de terrain. Il existe également un précédent dans l'industrie d'automation consistant à (correctement) utiliser le terme passerelle pour identifier des appareils qui convertissent un protocole de bus de terrain sans fil en un protocole de bus de terrain câblé pour le rattachement à un système de commande. Le présent document adhère à l'usage de l'industrie d'automation dans une tentative de réduire au maximum la confusion.

[18] L'objet tunnel natif utilise les services natifs de tunnel et de publication pour transporter un contenu de message étranger. Un appareil de terrain qui requiert le contenu de message étranger pour accomplir sa fonction ne peut pas être considéré comme étant un appareil natif.

La présente norme fournit la fonctionnalité de prise en charge pour la construction de passerelles. Elle ne fournit pas les détails complets sur la façon de construire une passerelle particulière quelconque. L'Annexe U est strictement informative, car aucune passerelle n'est spécifiée. À ce titre, elle fournit une base suggérée pour la future construction d'une spécification de passerelles, mais elle n'en est pas une elle-même. Aucune validation du contenu de l'Annexe U n'a eu lieu.

L'Annexe U décrit la fonctionnalité de prise en charge pour les besoins de conversion de protocoles étrangers, mais elle ne décrit pas en détails la façon d'accomplir une conversion spécifique de protocoles ou la façon d'interfacer à un quelconque réseau d'installation.

Les protocoles hérités n'ont pas été conçus pour opérer sur des réseaux sans fil. Ils n'accèdent pas à l'information d'une manière qui conserve l'énergie, et ils sont souvent intolérants à l'accès différé aux appareils au repos qui conservent l'énergie. La fonctionnalité de prise en charge de passerelle de la présente norme est, en grande partie, prévue pour permettre la construction de passerelles qui adaptent des protocoles hérités aux conditions des appareils sans fil à faible consommation d'énergie.

Le rôle de passerelle inclut un UAP spécialisé. La fonctionnalité est fournie par des objets d'AL et le fonctionnement interne de passerelles. Les objets utilisent les interfaces de la suite de protocoles de communication pour prendre en charge des fonctions d'interface de côté haut des passerelles.

U.1.2 Diagrammes hypothétiques des suites de protocoles de passerelles pour appareils et adaptateurs natifs

Le diagramme à la Figure 17 montre une passerelle hypothétique interfaçant une application de niveau hôte (le système de commande exemplaire) à un appareil de terrain sans fil. Dans ce cas, l'appareil de terrain est un appareil natif. La conversion de protocole est accomplie dans la passerelle pour convertir entre le protocole de réseau d'installation et le protocole natif de la présente norme. Les routeurs peuvent exister entre la passerelle et l'appareil de champ (voir Figure 17), mais du point de vue de la passerelle, leur opération est transparente.

Le diagramme à la Figure 19 montre une passerelle hypothétique interfaçant une application de niveau hôte (le système de commande exemplaire) à un appareil E/S par un système sans fil qui se conforme à la présente norme. Dans ce cas spécifique, l'appareil E/S interfacé est un appareil hérité, pas un appareil sans fil natif, et un adaptateur est donc exigé. La conversion de protocole est accomplie à la fois dans la passerelle et dans l'adaptateur. La passerelle et l'adaptateur convertissent chacun entre les protocoles hérités et les protocoles de transmission spécifiés par la présente norme.

NOTE 1 Il est souvent possible de mettre en œuvre un adaptateur à un seul appareil E/S câblé hérité en accomplissant une conversion de protocole vers et depuis des formats natifs, sans transporter un quelconque contenu de message étranger. À une passerelle, une telle combinaison d'un adaptateur et d'un appareil hérité connecté est indiscernable d'un appareil E/S natif. Aucune disposition de passerelle spéciale n'est adoptée pour de tels appareils. En plus, il n'y a aucune disposition de passerelle spéciale de faciliter le multiplexage d'un tel adaptateur à plusieurs appareils E/S câblés hérités.

Comme montré à la Figure 19, une passerelle hypothétique et un adaptateur hypothétique partagent une structure commune. Ils ont tous les deux une interface et une suite de protocoles pour un réseau étranger. Ils ont tous les deux une interface et une suite de protocoles telles que décrites dans la présente norme. Ils ont tous les deux des convertisseurs de protocoles. La structure commune s'étend même davantage – ils peuvent partager des objets communs et une structure commune d'interface de côté haut. Pour cette raison, aucun rôle séparé n'a été décrit pour un adaptateur.

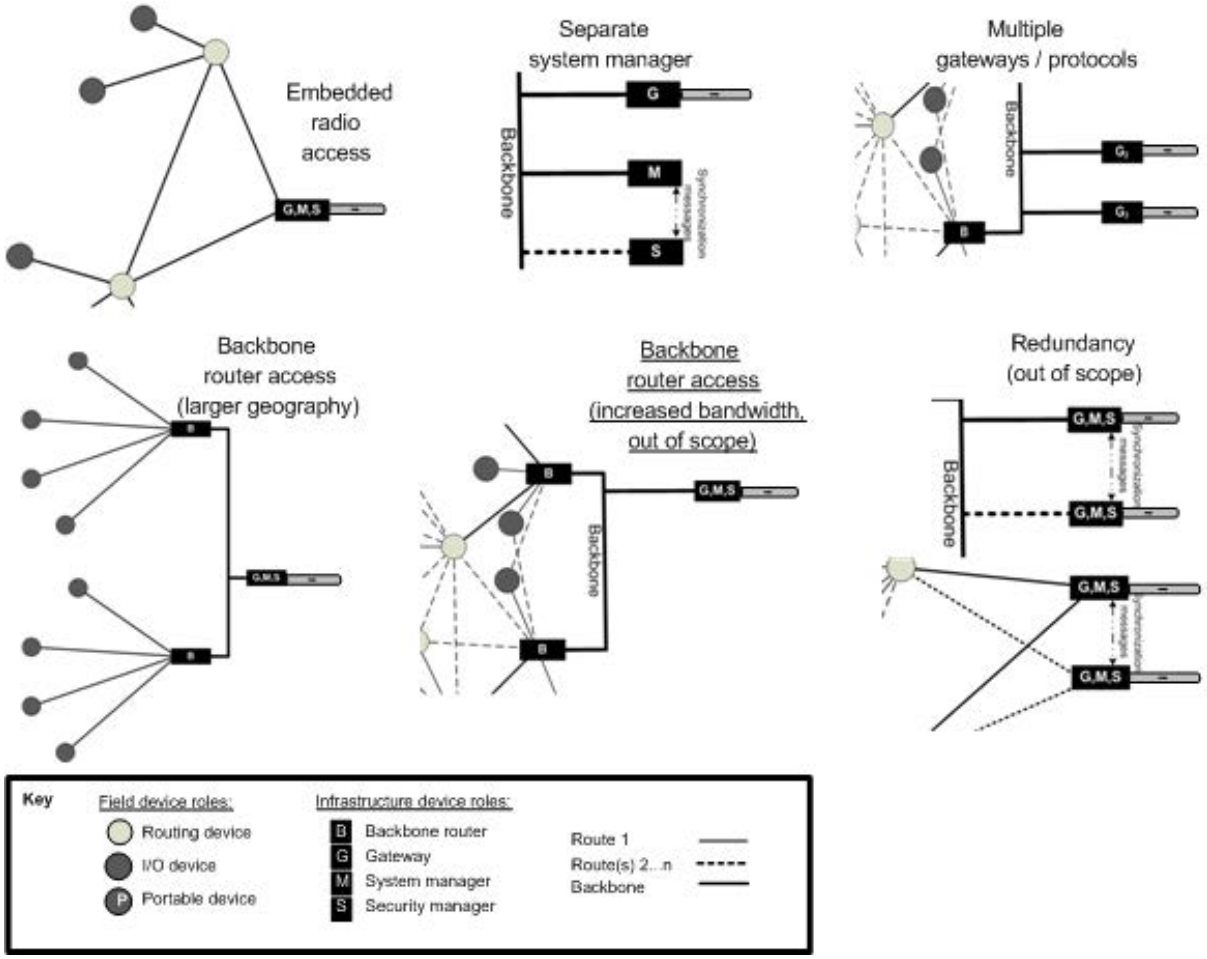
NOTE 2 Les différences entre une passerelle et un adaptateur se rapportent surtout à la mise en œuvre. Par exemple, certains protocoles hérités éditent seulement à partir du terrain, ce qui requiert la prise en charge de la fonctionnalité de producteur, mais pas la fonctionnalité de consommateur, dans l'adaptateur. D'autres protocoles hérités prennent également en charge l'édition vers le terrain et, donc, exigent à la fois la fonctionnalité de producteur et celle de consommateur. Dans un autre exemple, des outils d'étude hérités transportés sur le terrain

et branchés au réseau hérité derrière l'adaptateur ont parfois besoin d'utiliser les mêmes fonctions que s'ils étaient derrière la passerelle.

Pour qu'une passerelle et un adaptateur soient interopérables, ils exigent la conversion de protocole commun. Si l'adaptateur convertit vers et depuis un format natif, la passerelle peut faire la même chose. Si l'adaptateur tunnellise un protocole hérité, la passerelle peut tunnelliser le même protocole.

U.1.3 Scénarios de passerelles

Des scénarios communs de passerelle sont montrés à la Figure U.1. Cette figure ne tente pas de fournir une description exhaustive de toutes les variantes; elle est plutôt incluse pour montrer les limites de la présente norme.



Légende

Anglais	Français
Embedded radio access	Accès radio intégré
Separate system manager	Gestionnaire de système distinct
Backbone	Dorsale
Synchronization messages	Messages de synchronisation
Multiple gateways / protocols	Plusieurs passerelles / protocoles
Backbone router access (larger geography)	Accès au routeur dorsal (géographie plus vaste)
Backbone router access (increased bandwidth, out of scope)	Accès routeur dorsal (largeur de bande accrue. hors domaine)
Redundancy (out of scope)	Redondance (hors domaine)
Key	Légende

Anglais	Français
Field device roles	Rôles d'appareil de terrain
Routing device	Appareil de routage
I/O device	Appareil E/S
Portable device	Appareil portatif
Infrastructure device roles	Rôles d'appareil d'infrastructure
B Backbone router	B Routeur dorsal
G Gateway	G Passerelle
M System manager	M Gestionnaire système
S Security manager	S Gestionnaire de sécurité
Route 1	Chemin 1
Route(s) 2…n	Chemin(s) 2…n
Backbone	Dorsale

Figure U.1 – Scénarios de passerelles

Telle que décrite à l'Article 5, une passerelle met en œuvre un rôle au sein du système. Une diversité de mises en œuvre physique est possible.

Certaines mises en œuvre d'appareil peuvent avoir une interface sans fil de Classe A et une pile de protocoles intégrés dans le même paquetage que la passerelle, fournissant l'accès direct au réseau sans fil. De façon indépendante, des mises en œuvre peuvent avoir des rôles de gestion de système et de sécurité qui coexistent avec le rôle de passerelle.

Les rôles de gestionnaire de système et de gestionnaire de sécurité peuvent ne pas être corésidents avec le rôle de passerelle. La passerelle n'interagit pas directement avec le gestionnaire de sécurité, mais indirectement par l'intermédiaire du gestionnaire de système. La fonctionnalité de passerelle exige un chemin de communication avec un gestionnaire de système pour fonctionner comme partie intégrante d'un système opérationnel.

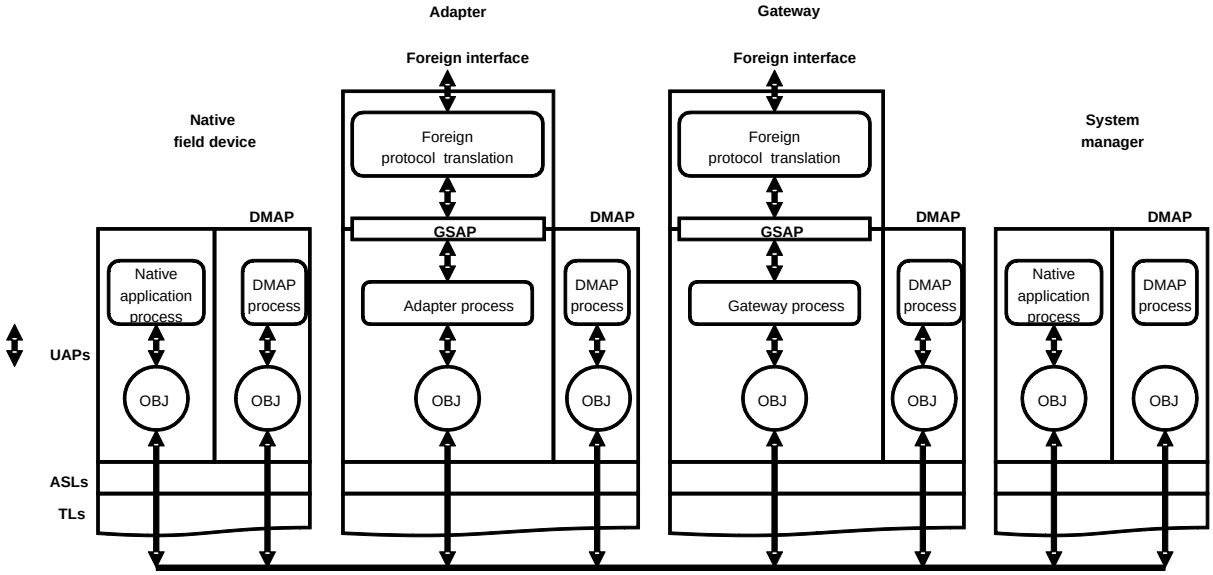
Un appareil mettant en œuvre un rôle de passerelle peut utiliser des routeurs dorsaux pour communiquer avec des appareils de terrain sans fil conformes à la présente norme.

La communication de passerelle avec des appareils de terrain via des routeurs dorsaux est de fonctionnement transparent. Des extensions de NL existent dans d'autres parties de la présente norme pour prendre en charge cette transparence. Il est toutefois nécessaire de configurer ce routage au sein de la passerelle et des routeurs dorsaux. Les routeurs dorsaux peuvent être utilisés pour étendre la portée géographique des appareils connectés par une passerelle. Des routeurs dorsaux peuvent également être utilisés pour augmenter la largeur de bande ou pour ajouter des chemins redondants entre une passerelle et un maillage.

Plusieurs passerelles indépendantes peuvent exister sur un système. Cela est facilité par un adressage et des relations de communication indépendants entre appareils. Une utilisation pour les passerelles indépendantes consiste à prendre en charge plusieurs protocoles. Aucune disposition spéciale n'est prise dans la présente norme pour le fonctionnement interdépendant entre passerelles, comme pour la redondance ou le partage de charge.

U.1.4 Modèle de passerelle de base

Les passerelles peuvent suivre le modèle général montré à la Figure U.2.



Légende

Anglais	Français
Native field device	Appareil de terrain natif
Native application process	Processus d'application natif
DMAP process	Processus de DMAP
Adapter	Adaptateur
Foreign interface	Interface étrangère
Foreign protocol translation	Conversion de protocoles étrangers
Adapter process	Processus d'adaptateur
Gateway	Passerelle
Gateway process	Processus de passerelle
System manager	Gestionnaire de système

Figure U.2 – Modèle de passerelle de base

Dans cet exemple, les passerelles et les adaptateurs hébergent des convertisseurs de protocoles étrangers qui reçoivent et émettent des messages d'interface étrangers (habituellement issus d'un système de commande, d'un système de gestion d'actifs, ou d'un système d'ingénierie) et utilisent des interfaces de passerelle pour interagir avec des appareils sans fil. Les interfaces de passerelle sont fournies par l'intermédiaire d'une interface de côté haut qui est atteinte en un GIAP. Les convertisseurs de protocoles locaux à un appareil utilisent ces interfaces par le GIAP.

La conversion de protocoles achemine les informations d'application pour la commande, la surveillance, la configuration, et la gestion. Les messages de protocole étrangers contiennent ces informations. La tunnellisation, la communication d'application de protocole étranger (FPAC), et des méthodes d'accès aux objets natifs sont fournies dans les objets décrits dans la présente norme pour prendre en charge la conversion de protocoles, telles que décrites en Annexe N. Chaque méthode nécessite des compromis spécifiques d'effort de conversion, d'efficacité énergétique et de performances. Les convertisseurs de protocoles pratiques sont susceptibles d'utiliser une combinaison de ces méthodes.

Les passerelles et les adaptateurs peuvent chacun avoir des processus d'application qui s'interfacent aux convertisseurs de protocoles par l'intermédiaire du GIAP. Chaque processus fournit les interfaces de côté haut en utilisant des objets d'application (OBJ). La communication entre objets utilise les méthodes de messagerie fournies par l'intermédiaire de la sous-couche d'application (ASL).

Les interfaces de GIAP sont utilisées pour la gestion de réseau, la tunnellisation de protocoles, le chargement et le téléchargement, les alertes, la gestion du temps, et l'accès aux objets natifs d'application et de gestion. Le GIAP est décrit en détail à l'Article U.2.

Pour les appareils d'automation câblés, un adaptateur accomplit une fonction symétrique et convertit des messages d'interface étrangers vers une interface côté haut d'adaptateur (aussi GIAP) par l'intermédiaire d'un convertisseur de protocole homologue de passerelle.

L'accès de passerelle à un appareil de terrain natif (à partir d'une passerelle) est activé par le même GIAP; cependant, les interactions d'objet utilisent la messagerie native exclusivement. Un processus d'application natif interagit avec le processus de passerelle ou d'adaptateur d'une manière qui dépend du type de l'objet.

Les passerelles, les adaptateurs et les appareils de terrain natifs peuvent être gérés par la même méthode symétrique. Le processus DMAP est le processus homologue dans ce cas. Cette gestion est spécifique à la présente norme et pas nécessairement aux protocoles étrangers. Ceux-ci peuvent fournir des méthodes complémentaires de gestion d'appareil et de bus de terrain câblé qui ne relèvent pas du domaine d'application de la présente norme.

Le modèle de base de passerelle inclut l'interfaçage au gestionnaire de système, qui permet au gestionnaire de système d'être accessible par l'intermédiaire de la passerelle.

Les routeurs dorsaux et les appareils de routage ne sont pas montrés à la Figure U.2, parce que la fonctionnalité de passerelle se produit au sein la couche d'application, pas au sein des couches de communication inférieures.

U.2 GIAP hypothétique

U.2.1 Résumé des interfaces et des primitives

La partie passerelle de la présente norme décrit un GIAP hypothétique qui peut servir d'interface de côté haut au-dessus d'une suite de protocoles de communication sans fil pour acheminer des informations sans fil et gérer le comportement sans fil. Ce GIAP hypothétique est générique et pourrait être utilisé comme une interface commune au-dessus de l'AL de la présente norme et au-dessus d'une autre fonctionnalité dans de semblables suites de protocoles de communication. L'Annexe P décrit une mise en œuvre potentielle des interfaces de GIAP pour la suite de protocoles sans fil de la présente norme, en utilisant les objets et interfaces d'AL définis. L'Annexe Q décrit une autre mise en œuvre d'interface de GIAP hypothétique pour une variante de suite de protocoles sans fil.

NOTE 1 Une intention première des interfaces de GIAP exemplaires est de permettre l'accès multimode où plusieurs interfaces sans fil sont disponibles à la passerelle. Dans les configurations où le chemin allant du gestionnaire de système au réseau d'installation est seulement par l'intermédiaire du rôle de passerelle, le GIAP prend en charge des informations de comptes-rendus cohérentes relatives à chaque interface sans fil sous-jacente pour promouvoir une coexistence améliorée. Des informations de gestion de système peuvent être incluses dans un rapport relatif aux performances de communication pour identifier des problèmes potentiels d'interférence, dans un rapport relatif à la topologie pour identifier les appareils contigus, et dans un rapport relatif aux informations de voies et de programmes pour identifier des conflits potentiels d'utilisation.

NOTE 2 Une autre intention du GIAP exemplaire est de fournir un modèle pour la configuration et l'accès de plusieurs réseaux sans fil sous-jacents. Cela réduit potentiellement l'effort pour les développeurs de passerelles s'ils ont plusieurs protocoles de bus de terrain à prendre en charge. Les interfaces de GIAP peuvent ou peuvent ne pas être applicables à des développeurs de passerelles spécialisées, tels que ceux qui servent un seul protocole étranger; les passerelles spécialisées peuvent préférer utiliser des interfaces internes personnalisées.

Cette interface de GIAP hypothétique est utilisable par une diversité de convertisseurs de protocoles pour s'interfacer à des suites de protocoles de communication sans fil pour acheminer des informations sans fil et gérer le comportement sans fil. Un convertisseur de protocole existe dans une passerelle. En fonction de la mise en œuvre, un convertisseur de protocoles et un GIAP peuvent également exister dans un adaptateur. Les convertisseurs de protocoles auront une complexité variable en fonction du protocole qui existe au-dessus de la passerelle et en dessous des adaptateurs. Certains protocoles utiliseront un sous-ensemble

de ces interfaces de GIAP hypothétiques. Par exemple, un protocole peut seulement exiger l'interaction C/S et ne pas exiger des interfaces P/S. Fonctionellement, un adaptateur est considéré comme étant un sous-ensemble d'une passerelle et serait seulement censé prendre en charge un sous-ensemble de ces interfaces de GIAP hypothétiques relatives à l'acheminent d'informations sans fil.

NOTE 3 Ce débat relatif aux passerelles ne décrit pas la conversion de protocoles pour des protocoles de fieldbus spécifiques.

Les interfaces de GIAP sont résumées dans le Tableau 1.

Tableau 1 – Résumé des exemples d'interfaces de côté haut de passerelles hypothétiques

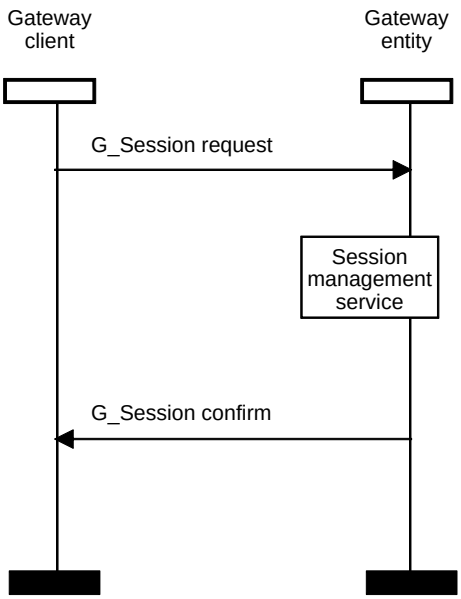
Exemple d'interface	Sous-type d'interface	Primitive	Description
Session	-	G_Session request G_Session confirm	Un convertisseur de protocoles étrangers au sein de la passerelle peut établir des sessions pour le compte de clients distants
En leasing	-	G_Lease request G_Lease confirm	Les locations permettent à la passerelle de gérer en interne ses ressources de communication internes sur une base par session
Device_List_Report	-	G_Device_List_Report request G_Device_List_Report confirm	Détermine les appareils associés au rôle de passerelle
Topology_Report	-	G_Topology_Report request G_Topology_Report confirm	Fournit un rapport de topologie relatif aux appareils dans un maillage sans fil. Cette interface peut être utile si, par exemple, le rôle de la passerelle fonctionne dans un système dans lequel l'interface de gestion du système vers le réseau de l'usine se fait via le rôle de la passerelle
Schedule_Report	-	G_Schedule_Report request G_Schedule_Report confirm	Fournit des allocations détaillées d'intervalles de temps et de voies sur une base par appareil. Cette interface peut être utile si, par exemple, le rôle de la passerelle fonctionne dans un système dans lequel l'interface de gestion du système vers le réseau de l'usine se fait via le rôle de la passerelle
Device_Health_Report	-	G_Device_Health_Report request G_Device_Health_Report confirm	Le rapport de santé d'appareil pour des appareils associés à la passerelle. Cette interface peut être utile si, par exemple, le rôle de la passerelle fonctionne dans un système dans lequel l'interface de gestion du système vers le réseau de l'usine se fait via le rôle de la passerelle

Exemple d'interface	Sous-type d'interface	Primitive	Description
Neighbor_Health_Report	-	G_Neighbor_Health_Report request	Le rapport de santé de communication pour l'ensemble d'appareils voisins associés à un appareil spécifique qui est associé à la passerelle. Cette interface peut être utile si, par exemple, le rôle de la passerelle fonctionne dans un système dans lequel l'interface de gestion du système vers le réseau de l'usine se fait via le rôle de la passerelle
		G_Neighbor_Health_Report confirm	
Network_Health_Report	-	G_Network_Health_Report request	Résumé de rapport de santé de communication pour le réseau sans fil. Cette interface peut être utile si, par exemple, le rôle de la passerelle fonctionne dans un système dans lequel l'interface de gestion du système vers le réseau de l'usine se fait via le rôle de la passerelle
		G_Network_Health_Report confirm	
Durée	-	G_Time request	Récupération et réglage du temps pour le réseau sans fil associé à la passerelle
		G_Time confirm	
Client/server	-	G_Client_Server request	Fournit la communications client/serveur
		G_Client_Server indication	
		G_Client_Server response	
		G_Client_Server confirm	
Publish/Subscribe (éditer/s'abonner)	Publish	G_Publish request	Fournit la communication éditeur/abonné
		G_Publish indication	
		G_Publish confirm	
	S'abonner	G_Subscribe request	
		G_Subscribe confirm	
	Publish_Timer	G_Publish_Timer indication	
	Subscribe_Timer	G_Subscribe_Timer indication	
	Watchdog_Timer	G_Watchdog_Timer indication	
Bulk_Transfer [a]	Ouvert	G_Bulk_Open request	Autorise le téléchargement montant et le téléchargement descendant d'éléments volumineux tels que des images de firmware et des tampons d'échantillons
		G_Bulk_Open confirm	
	Transfert	G_Bulk_Transfer request	
		G_Bulk_Transfer confirm	
	Close	G_Bulk_Close request	
		G_Bulk_Close confirm	

Exemple d'interface	Sous-type d'interface	Primitive	Description
Alerte	S'abonner	G_Alert_Subscription request	Autorise l'abonnement et la réception d'alertes spécifiques
		G_Alert_Subscription confirm	
	Notify (notifier)	G_Alert_Notification indication	
Gateway_Configuration	Read	G_Read_Gateway_Configuration request	Fournit l'accès en lecture et écriture à des attributs de configuration de la passerelle
		G_Read_Gateway_Configuration confirm	
	Write	G_Write_Gateway_Configuration request	
		G_Write_Gateway_Configuration confirm	
Device_Configuration	Read	G_Read_Device_Configuration request	Autorise la passerelle à déterminer quels appareils lui sont associés
		G_Read_Device_Configuration confirm	
	Write	G_Write_Device_Configuration request	
		G_Write_Device_Configuration confirm	
a Les primitives d'interface sont communes aux opérations de téléchargement montant et descendant.			

U.2.2 Séquence de primitives

La Figure U.3, la Figure U.4, la Figure U.5, la Figure U.6, la Figure U.7, la Figure U.8, la Figure U.9, la Figure U.10, la Figure U.11, la Figure U.12, la Figure U.13, la Figure U.14 et la Figure U.15 montrent les séquences de primitives pour les interfaces de côté haut de passerelles. Les figures sont décrites en termes de client interne à une passerelle, d'entité de passerelle, de client d'appareil, et d'entité d'appareil. Un client interne à une passerelle est un utilisateur des interfaces de GIAP au sein d'une passerelle. Une entité de passerelle est un fournisseur d'interfaces de GIAP au sein de la passerelle. La fourniture d'interfaces nécessite des interactions complémentaires à travers le réseau sans fil à un ou plusieurs appareils. Un client d'appareil est un utilisateur d'interfaces de GIAP au sein d'un appareil. Une entité d'appareil est un fournisseur d'interfaces de GIAP au sein de l'appareil.

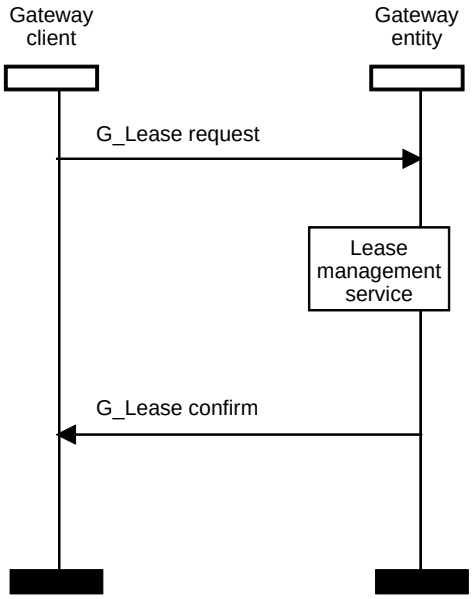


Légende

Anglais	Français
Gateway client	Client de passerelle

Anglais	Français
Gateway entity	Entité de passerelle
G_session request	Demande de G_session
Session management service	Service de gestion de session
G_session confirm	Confirmation de G_session

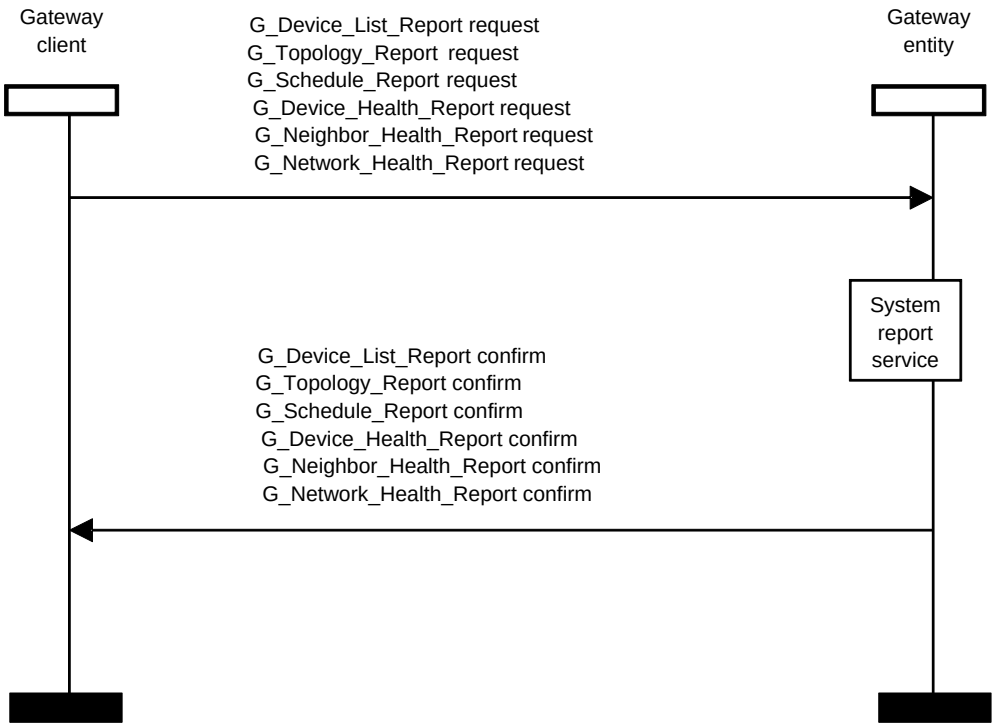
Figure U.3 – Séquence interne de primitives pour interface de session



Légende

Anglais	Français
Gateway client	Client de passerelle
Gateway entity	Entité de passerelle
G_Lease request	Demande de G_Lease
Lease management service	Service de gestion de location
G_Lease confirm	Confirmation de G_Lease

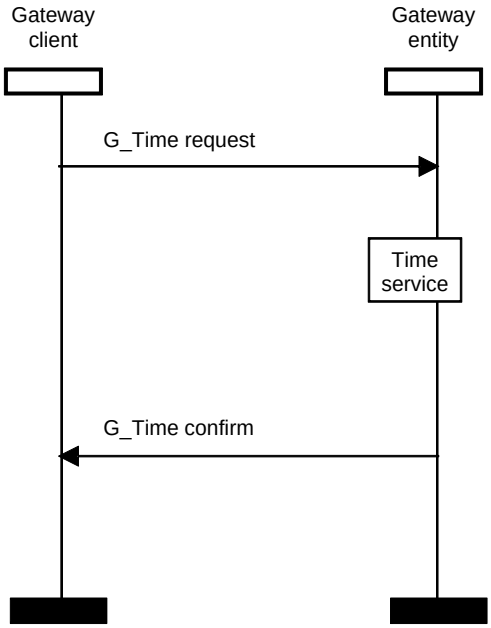
Figure U.4 – Séquence interne de primitives pour interface de gestion de locations



Légende

Anglais	Français
Gateway client	Client de passerelle
Gateway entity	Entité de passerelle
G_Device_List_Report request	Demande de G_Device_List_Report
G_Topology_Report request	Demande de G_Topology_Report
G_Schedule_Report request	Demande de G_Schedule_Report
G_Device_Health_Report request	Demande de G_Device_Health_Report
G_Neighbour_Health_Report request	Demande de G_Neighbour_Health_Report
G_Network_Health_Report request	Demande de G_Network_Health_Report
G_Device_List_Report confirm	Confirmation de G_Device_List_Report
G_Topology_Report confirm	Confirmation de G_Topology_Report
G_Schedule_Report confirm	Confirmation de G_Schedule_Report
G_Device_Health_Report confirm	Confirmation de G_Device_Health_Report
G_Neighbour_Health_Report confirm	Confirmation de G_Neighbour_Health_Report
G_Network_Health_Report confirm	Confirmation de G_Network_Health_Report
System report service	Service de rapports système

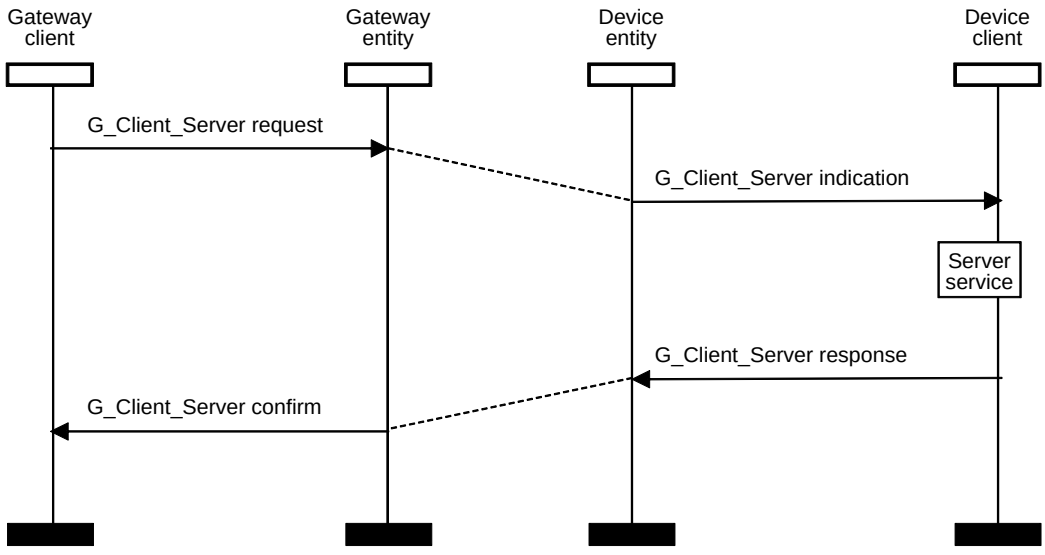
Figure U.5 – Séquence interne de primitives pour interface de rapport système



Légende

Anglais	Français
Gateway client	Client de passerelle
Gateway entity	Entité de passerelle
G_Time request	Demande de G_Time
Time service	Service de temps
G_Time confirm	Confirmation de G_Time

Figure U.6 – Séquence interne de primitives pour interface de session

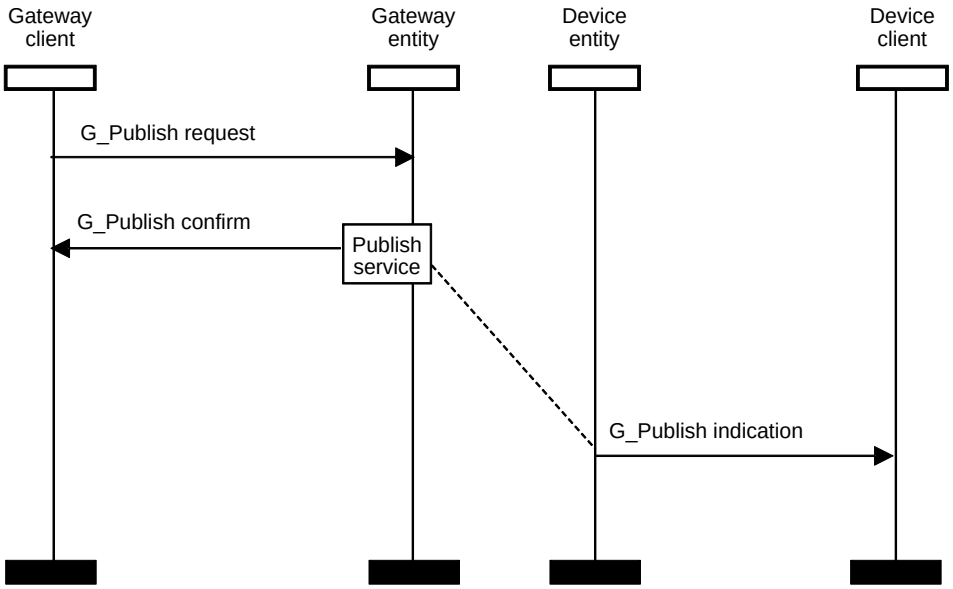


Légende

Anglais	Français
Gateway client	Client de passerelle
Gateway entity	Entité de passerelle
Device entity	Entité d'appareil
Device client	Client d'appareil
G_Client_Server request	Demande de G_Client_Server
G_Client_Server indication	Indication de G_Client_Server indication
Server service	Service serveur

Anglais	Français
G_Client_Server confirm	Confirmation de G_Client_Server
G_Client_Server response	Réponse de G_Client_Server

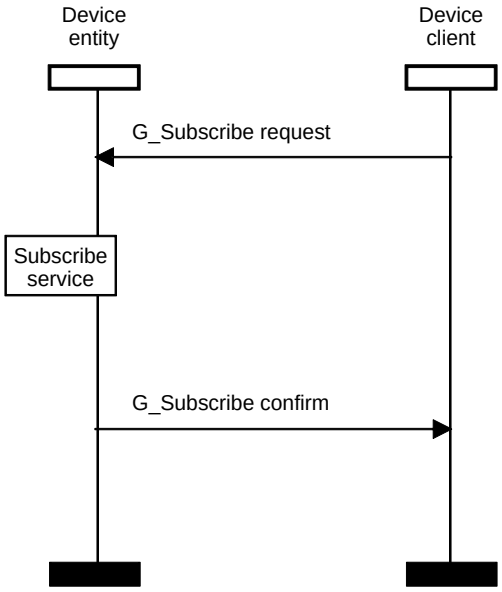
Figure U.7 – Séquence interne de primitives pour interface client/serveur initiée d'une passerelle à un appareil adaptateur



Légende

Anglais	Français
Gateway client	Client de passerelle
Gateway entity	Entité de passerelle
Device entity	Entité d'appareil
Device client	Client d'appareil
G_Publish request	Demande de G_Publish
G_Publish confirm	Confirmation de G_Publish
Publish service	Service Publish
G_Publish indication	Indication de G_Publish

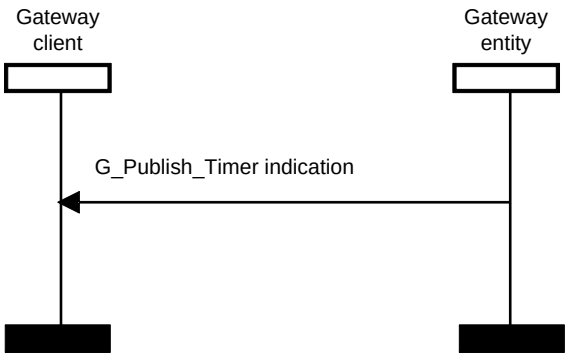
Figure U.8 – Séquence interne de primitives pour interface Publish initiée d'une passerelle à un appareil adaptateur



Légende

Anglais	Français
Device entity	Entité de passerelle
Device client	Client de passerelle
G_Suscribe request	Demande de G_Subscribe
Subscribe service	Service Subscribe
G_Subscribe confirm	Confirmation de G_ Subscribe

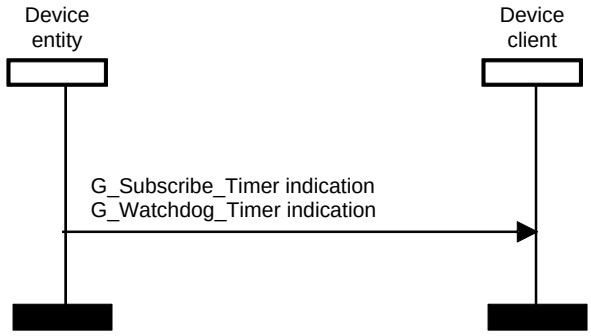
Figure U.9 – Séquence interne de primitives pour interface Subscribe initiée à partir d'un appareil adaptateur



Légende

Anglais	Français
Gateway client	Client de passerelle
Gateway entity	Entité de passerelle
G_Publish_Timer indication	Indication de G_Publish_Timer

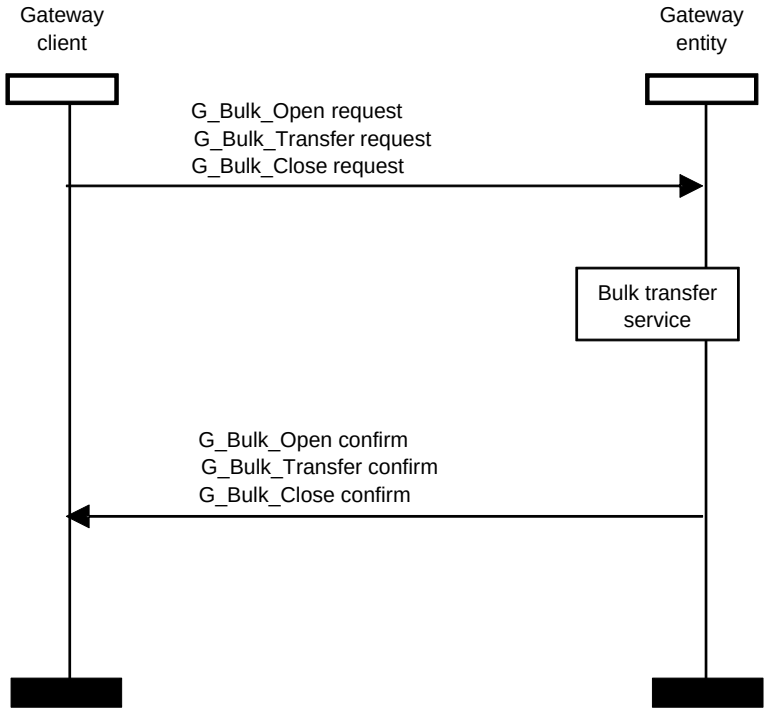
Figure U.10 – Séquence interne de primitives pour temporisateur d'éditeur initié d'une passerelle à un appareil adaptateur



Légende

Anglais	Français
Device entity	Entité de passerelle
Device client	Client de passerelle
G_Suscribe_Timer indication	Indication de G_Suscribe_Timer
G_Watchdog_Timer indication	Indication de G_Watchdog_Timer

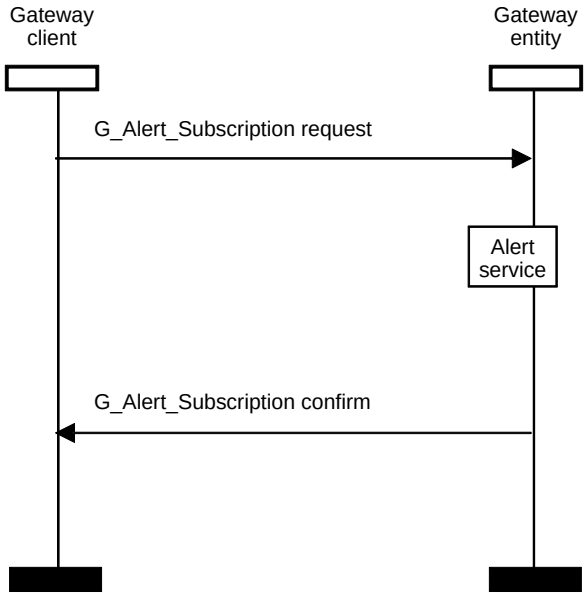
Figure U.11 – Séquence interne de primitives pour temporisateurs d'abonné initiée à partir d'un appareil adaptateur



Légende

Anglais	Français
Gateway client	Client de passerelle
Gateway entity	Entité de passerelle
G_Bulk_Open request	Demande de G_Bulk_Open
G_Bulk_Transfer request	Demande de G_Bulk_Transfer
G_Bulk_Close request	Demande de G_Bulk_Close
Bulk transfer service	Service de transfert en masse
G_Bulk_Open confirm	Confirmation de G_Bulk_Open
G_Bulk_Transfer confirm	Confirmation de G_Bulk_Transfer
G_Bulk_Close confirm	Confirmation de G_Bulk_Close

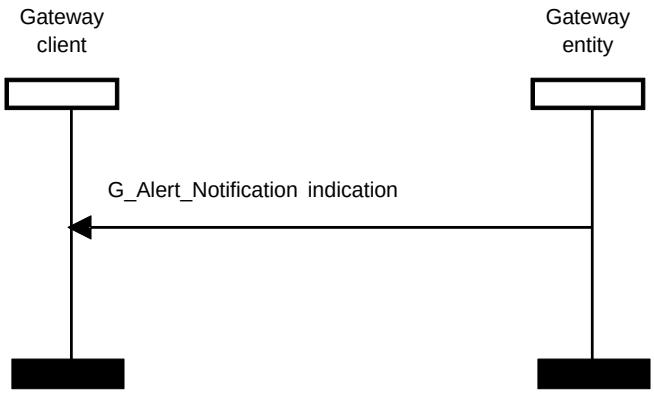
Figure U.12 – Séquence interne de primitives pour interface de transfert en masse



Légende

Anglais	Français
Gateway client	Client de passerelle
Gateway entity	Entité de passerelle
G_Alert_Subscription request	Demande de G_Alert_Subscription
G_Alert_Subscription confirm	Confirmation de G_Alert_Subscription
Alert service	Service Alert

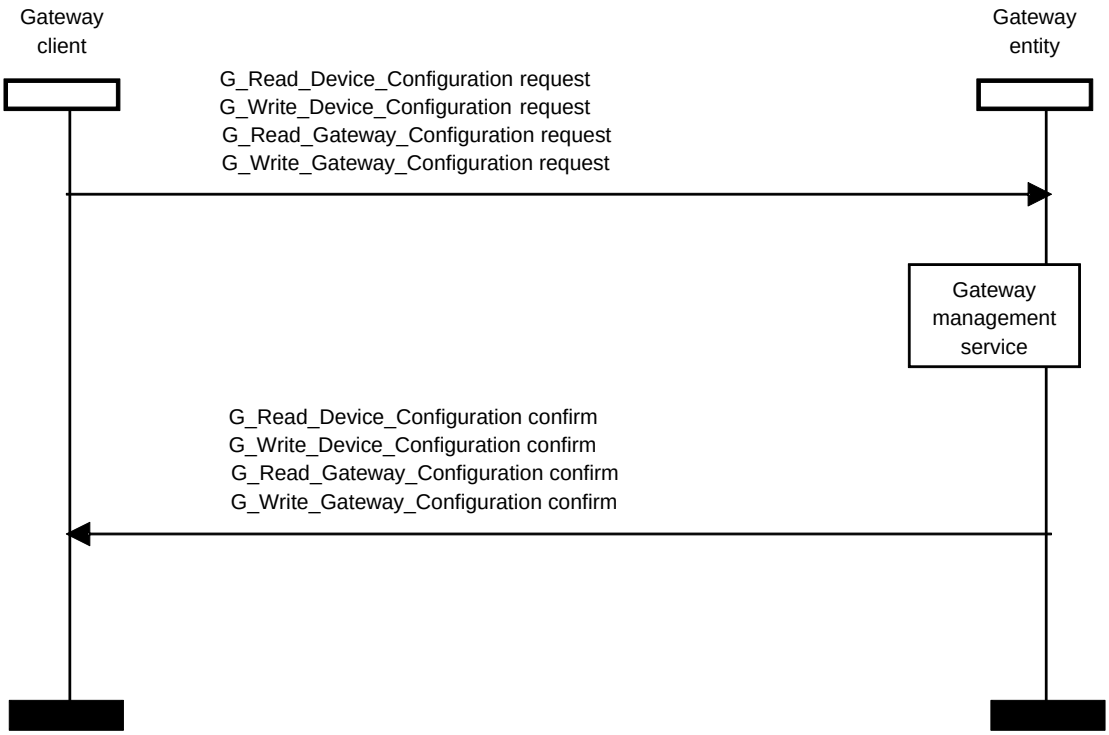
Figure U.13 – Séquence interne de primitives pour interface d'abonnement d'alertes



Légende

Anglais	Français
Gateway client	Client de passerelle
Gateway entity	Entité de passerelle
G_Alert_Notification indication	Indication de G_Alert_Notification

Figure U.14 – Séquence interne de primitives pour interface de notification d'alertes



Légende

Anglais	Français
Gateway client	Client de passerelle
Gateway entity	Entité de passerelle
G_Read_Device_Configuration request	Demande de G_Read_Device_Configuration
G_Write_Device_Configuration request	Demande de G_Write_Device_Configuration
G_Read_Gateway_Configuration request	Demande de G_Read_Gateway_Configuration
G_Write_Gateway_Configuration request	Demande de G_Write_Gateway_Configuration
Gateway management service	Service de gestion de passerelle
G_Read_Device_Configuration confirm	Confirmation de G_Read_Device_Configuration
G_Write_Device_Configuration confirm	Confirmation de G_Write_Device_Configuration
G_Read_Gateway_Configuration confirm	Confirmation de G_Read_Gateway_Configuration
G_Write_Gateway_Configuration confirm	Confirmation de G_Write_Gateway_Configuration

Figure U.15 – Séquence interne de primitives pour interface de gestion de passerelle

U.2.3 Description détaillée des paramètres

U.2.3.1 Généralités

Les paramètres qui sont communs à plusieurs interfaces sont décrits en U.2.3. Les paramètres qui sont uniques à une interface sont décrits au sein de l'interface en question.

NOTE Sachant que la présente norme ne définit aucune passerelle, et parce que le GIAP débattu est hypothétique, tous ces paramètres sont également strictement hypothétiques, tout comme le sont tous les énoncés à leur sujet.

U.2.3.2 Paramètre GS_Session_ID

Le paramètre GS_Session_ID identifie de façon univoque une session spécifique.

Un identificateur de session valide qui n'a pas expiré est fourni afin d'invoquer toutes les interfaces, à l'exception des interfaces de session.

U.2.3.3 Paramètre GS_Transaction_ID

Le paramètre GS_Transaction_ID identifie de façon univoque les parties demande et réponse d'une transaction au sein du contexte d'une session spécifique.

L'identificateur de transaction est utilisé pour apparier une demande d'interface de GIAP avec la réponse d'interface de GIAP correspondante.

U.2.3.4 Paramètre GS_Lease_ID

GS_Lease_ID identifie les ressources d'entité de passerelle et les ressources de communication qui sont allouées à une session particulière.

L'identificateur de location est fourni par l'utilisateur d'interface de GIAP lorsqu'une interface est invoquée pour identifier les ressources de communication particulières utilisées pour prendre en charge l'interface.

U.2.3.5 Paramètre GS_Status

GS_Status est retourné par une primitive "confirm". Il peut représenter soit le statut résultant du traitement d'une demande locale, soit le statut correspondant à une réponse reçue en provenance d'une entité distante.

Le statut indique le succès ou l'échec de l'appel d'interface et, le cas échéant, la raison de l'échec.

U.2.3.6 Paramètre GS_Network_Address

GS_Network_Address est une adresse IPv6 utilisée pour identifier un appareil logique qui est unique à travers tous les réseaux.

Ce paramètre identifie de façon univoque une NLE spécifique.

U.2.3.7 Paramètre GS_Unique_Device_ID

GS_Unique_Device_ID est une adresse EUI64.

Ce paramètre identifie de façon univoque un appareil physique spécifique pour des besoins de gestion d'actifs.

U.2.3.8 Paramètre GS_Network_ID

GS_Network_ID est un identificateur unique pour un de plusieurs réseaux qui peuvent être accessibles par l'intermédiaire d'une seule passerelle.

Ce paramètre identifie de façon univoque un réseau spécifique.

U.2.3.9 Paramètre GS_Time

GS_Time est champ de temps TAI de 48 bits.

Le paramètre de temps est utilisé pour décrire les champs relatifs au temps tels qu'un marqueur temporel, un temps de démarrage et un temps d'arrêt.

U.2.3.10 Paramètre GS_Transfer_Mode

GS_Transfer_Mode identifie des variantes de transfert au niveau d'une GPDU.

GS_Transfer_Mode est fourni avec chaque GPDU fournie pour le transfert afin d'indiquer la qualité de service désirée et la priorité associée au transfert des PDU générées pour prendre en charge la primitive d'interface hypothétique.

U.2.4 Description détaillée des interfaces

U.2.4.1 Interface de gestion de session

U.2.4.1.1 Généralités

Une entité de passerelle est un processus au sein d'une passerelle qui fournit des interfaces de passerelle par le GIAP. Un client interne à une passerelle est un utilisateur d'interfaces fournies par une entité de passerelle. Les clients types internes à une passerelle incluent des systèmes hôtes, des systèmes de gestion d'actifs, et des outils d'ingénierie.

Les entités de passerelle fournissent des interfaces aux clients internes à une passerelle au sein du contexte d'une session. L'interface de gestion de session est utilisée pour établir et gérer ces sessions. Toutes les autres interfaces fournies par une entité de passerelle sont utilisées dans le contexte d'une session établie.

Un convertisseur de protocoles étrangers au sein de la passerelle peut établir des sessions pour le compte de clients distants et accomplir une conversion de protocoles sur les flux de communication qui correspondent à des interfaces fournies par une entité de passerelle.

Le but premier d'une session est d'autoriser une allocation de ressources et la récupération en masse des ressources de passerelle et de communication sur une base par client d'entité de passerelle.

Une session peut être établie par un processus local ou à distance (comme par une session distante TCP/IP).

Une ou plusieurs sessions peuvent exister simultanément entre une entité de passerelle et un ou plusieurs clients internes à une passerelle. Chaque session est identifiée de façon univoque.

NOTE Le nombre de sessions simultanées prises en charge est dépendant d'une mise en œuvre. Il est possible que certaines réalisations fournissent à une passerelle de fonction fixe une seule session, alors que d'autres mises en œuvre fournissent un certain nombre de sessions qui sont allouées à la demande à une diversité d'applications, y compris des systèmes hôtes, des historiens, des outils de gestion d'actifs et des outils d'ingénierie.

Le client interne à une passerelle utilise la primitive G_Session pour créer, renouveler ou supprimer une session.

U.2.4.1.2 Primitive G_Session

U.2.4.1.2.1 Primitives et leurs paramètres

Le Tableau U.2 décrit l'utilisation des paramètres pour la primitive G_Session.

Tableau U.2 – Utilisation des paramètres de la primitive G_Session

Nom du paramètre	G_Session	
	Demande	Confirmation
GS_Session_ID	M	M
GS_Session_Period	M	M
GS_Network_ID	M	-
GS_Status	-	M

U.2.4.1.2.2 Utilisation de demande de G_Session

Le client interne à une passerelle utilise la primitive de demande de G_Session pour créer, renouveler ou supprimer une session.

Une session est créée en fournissant un identificateur de session null (GS_Session_ID = 0) et une durée de session demandée. La présentation de GS_Session_Period > 0 demande une session de durée limitée, spécifiée en secondes. La présentation de GS_Session_Period = -1 demande une durée de session indéfinie.

Une session de durée limitée est renouvelée en fournissant un identificateur de session existant (autre que null) et une durée de session supérieure à 0 s. Une session de durée indéfinie n'a pas besoin d'être renouvelée.

Changer une session de durée limitée en une session de durée indéfinie en tentant de la renouveler avec une durée spécifiée de -1 n'est pas autorisé.

NOTE La limite supérieure de la durée de session est dépendante de la mise en œuvre. Par exemple, des mises en œuvre sont capables de dédier des ressources à des applications spécifiques, telles qu'un système hôte, ne libérant jamais les ressources en question.

Une session est supprimée en fournissant un identificateur de session existant (autre que null) et une durée de session de 0 s.

Une passerelle peut se connecter à plusieurs réseaux. Chaque session est associée à un réseau spécifique. Un identificateur de réseau (GS_Network_ID) est spécifié pour établir un réseau particulier pour la session. La portée d'identificateurs supplémentaires utilisés au sein d'une session est limitée au réseau particulier.

U.2.4.1.2.3 Utilisation de confirmation de G_Session

L'entité de passerelle utilise la primitive de confirmation de G_Session pour parachever la demande de G_Session au client interne à une passerelle.

Pour une demande réussie de création de session, l'entité de passerelle retourne un identificateur de session unique et non null. Cet identificateur est utilisé dans les opérations ultérieures de renouvellement et de suppression de session. GS_Session_Period est retourné avec la réelle durée de session allouée par l'entité de passerelle.

La valeur de GS_Session_Period retournée peut ne pas être la même que la valeur demandée.

Pour une demande de renouvellement de session, l'identificateur de session de demande est repris en écho et une nouvelle durée de session est retournée.

Pour une demande de suppression de session, l'identificateur de session est repris en écho, et la durée de session est mise à 0 s.

GS_Status est retourné pour spécifier le succès ou l'échec de l'opération, conformément au Tableau U.3.

Tableau U.3 – GS_Status pour confirmation de G_Session

Valeur	Colonne
0	Réussite; nouvelle session créée, renouvelée ou supprimée
1	Réussite; nouvelle session créée ou renouvelée avec période réduite
2	Echec; la session n'existe pas pour renouvellement ou suppression
3	Echec; la session ne peut pas être créée (aucune session supplémentaire disponible) Cela peut se produire si, par exemple, des sessions ont expiré mais n'ont pas été explicitement supprimées
4	Echec; autre

U.2.4.2 Interface de gestion de locations

U.2.4.2.1 Généralités

Les entités de passerelle allouent des ressources de communication aux clients internes à une passerelle par l'intermédiaire de locations. L'interface de gestion de locations est utilisée pour établir et gérer des locations.

Le but premier d'une location est de permettre l'allocation et la récupération à grain fin de ressources de communication sur une base par session.

Des ressources peuvent être allouées séparément selon les besoins de communication. Par exemple, les ressources C/S, P/S, de transfert en masse et d'abonnement d'alertes peuvent être allouées séparément.

Une ou plusieurs locations peuvent exister simultanément entre une entité de passerelle et un ou plusieurs clients internes à une passerelle. Chaque location est identifiée de façon univoque au sein d'une session.

U.2.4.2.2 Utilisation de l'interface

Le client interne à une passerelle utilise la primitive G_Lease pour créer, renouveler ou supprimer une location.

U.2.4.2.3 Primitive G_Lease

U.2.4.2.3.1 Primitives et leurs paramètres

Le Tableau U.4 décrit l'utilisation des paramètres pour la primitive G_Lease.

Tableau U.4 – Utilisation des paramètres de la primitive G_Lease

Nom du paramètre	G_Lease	
	Demande	Confirmation
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Lease_ID	M	M
GS_Lease_Period	M	M
GS_Lease_Type	M	-
GS_Protocol_Type	M	-
GS_Network_Address_List	C	-
GS_Network_Address	C	-
GS_Resource	C	-
GS_Lease_Parameters	C	-
GS_Transfer_Mode	C	-
GS_Update_Policy	C	-
GS_Period	C	-
GS_Phase	C	-
GS_Stale_Limit	C	-
GS_Connection_Info	C	-
GS_Wireless_Parameters	C	-
GS_Status	-	M

U.2.4.2.3.2 Utilisation de demande de G_Lease

Le client interne à une passerelle utilise la primitive de demande de G_Lease pour créer, renouveler ou supprimer une location.

Un identificateur de session (GS_Session_ID) est inclus dans les primitives de demande de G_Lease.

Un identificateur de transaction unique de session (GS_Transaction_ID) est spécifié pour chaque invocation de l'interface.

Une location est créée en fournissant un identificateur de location null (GS_Lease _ID = 0) et une durée de location demandée. La présentation de GS_Lease_Period > 0 demande une location de durée limitée, spécifiée en secondes. La présentation de GS_Lease_Period = 0 demande une durée de location indéfinie.

Une location de durée limitée est renouvelée en fournissant un identificateur de location existant (autre que null) et une durée de location supérieure à 0 s. Une location de durée indéfinie n'a pas besoin d'être renouvelée. Une location de durée limitée ne peut pas être changée en une location de durée indéfinie par renouvellement avec une durée de 0 s.

La valeur prise en charge maximale est dépendante d'une mise en œuvre. Des mises en œuvre sont autorisées à choisir de dédier des ressources à des applications spécifiques, telles qu'un système hôte, ne libérant jamais les ressources en question.

Une location est supprimée en fournissant un identificateur de location existant et une durée de location demandée de 0 s.

Les différents types de locations sont disponibles, tels que spécifiés par GS_Lease_Type et montrés dans le Tableau U.5. Chaque type de location alloue des ressources d'entité de passerelle spécifiques à une location et des ressources de communication pour le compte du client interne à une passerelle.

Tableau U.5 – GS_Lease_Type pour demande de G_Lease

Valeur	Colonne
0	Client
1	Serveur
2	Editeur
3	Abonné
4	Client de transfert en masse
5	Serveur de transfert en masse
6	Abonnement d'alerte

GS_Protocol_Type identifie le protocole qui est associé à la location, tel que spécifié dans l'Annexe M. La spécification du type de protocole permet le traitement spécial pour des protocoles particuliers au sein de l'entité de passerelle.

Toutes les locations se rapportent à l'établissement des interfaces de communication entre l'entité de passerelle et un ou plusieurs appareils spécifiés par un ou plusieurs éléments dans GS_Network_Address_List. Les locations d'abonnement d'alertes n'allouent pas de ressources de communication au cours de l'établissement de location, mais le font dynamiquement à mesure que les abonnements alertes sont modifiés. GS_Network_Address_List n'est pas utilisé avec le type de location d'abonnement d'alertes.

Client, serveur, abonné, client de transfert en masse et serveur de transfert en masse décrivent seulement un seul élément au sein de GS_Network_Address_List. Le type de location d'éditeur peut décrire plusieurs éléments.

La spécification de plusieurs adresses IPv6 au sein de GS_Network_Address_List représente un groupe de multidiffusion. Les éléments au sein de G_Network_Address_List incluent GS_Network_Address.

GS_Lease_Parameters est une structure de paramètres avec la spécification des paramètres nécessaires pour l'établissement de certains types de location. L'utilisation de GS_Lease_Parameters est conditionnée au type de location spécifique comme suit.

NOTE L'Annexe P fournit des informations complémentaires relatives à l'utilisation détaillée de GS_Lease_Parameters.

Les locations client, serveur, éditeur, et abonné décrivent un GS_Resource unique. Cette valeur identifie des points d'extrémité de connexion client et serveur concordants et elle identifie également des points d'extrémité éditeur et abonné concordants. GS_Resource est également spécifié par le client de transfert en masse pour identifier l'élément de téléchargement montant/téléchargement descendant.

Les locations éditeur exigent une spécification de GS_Update_Policy, de GS_Period, de GS_Phase et de GS_Stale_Limit pour commander la temporisation et le comportement tamponné. GS_Transfer_Mode est également spécifié afin d'établir la qualité de transfert par défaut de l'interface et la priorité.

Les locations abonnées exigent une spécification de GS_Update_Policy, de GS_Period, de GS_Phase et de GS_Stale_Limit pour commander la temporisation et le comportement tamponné.

Un éditeur et un abonné peuvent accepter de décrire GS_Connection_Info dans la location abonnée pour la mise à disposition sur chaque réception de publication.

Les locations client et serveur décrivent GS_Transfer_Mode afin d'établir la qualité de transfert par défaut de l'interface et la priorité.

Une utilisation complémentaire du champ GS_Wireless_Parameters dépend de la construction de la passerelle. Cela permet l'accès à toutes les caractéristiques de communication qui sont exposées et peuvent être demandées.

U.2.4.2.3.3 Utilisation de confirmation de G_Lease

L'entité de passerelle utilise la primitive G_Lease pour parachever la demande de G_Lease au client interne à une passerelle.

L'identificateur de session (GS_Session_ID) et l'identificateur de transaction (GS_Transaction_ID) sont retournés pour permettre l'appariement de la primitive de confirmation avec la primitive de demande d'origine.

Pour une demande réussie de création de session, l'entité de passerelle retourne un identificateur de location unique de session. Cet identificateur de liaison est utilisé dans les opérations ultérieures de renouvellement et de suppression de location. GS_Lease_Period est retourné avec la réelle durée de location allouée par l'entité de passerelle.

Pour une demande de renouvellement de location, l'identificateur de location de demande est repris en écho et la durée de location réelle est donnée.

Pour une demande de suppression de location, l'identificateur de location est repris en écho, et la durée de location est mise à 0 s.

GS_Status est retourné pour spécifier le succès ou l'échec de l'opération, conformément au Tableau U.6.

Tableau U.6 – GS_Status pour confirmation de G_Lease

Valeur	Colonne
0	Réussite; nouvelle location créée, renouvelée ou supprimée
1	Réussite; nouvelle location créée ou renouvelée avec période réduite
2	Echec; la location n'existe pas pour renouvellement ou suppression
3	Echec; aucune location supplémentaire disponible
4	Echec; aucun appareil n'existe à l'adresse IPv6
5	Echec; type de location non valide
6	Echec; information de type de location non valide
7	Echec; autre

U.2.4.3 Interface de rapports de liste d'appareils

U.2.4.3.1 Généralités

L'interface de rapports de liste d'appareils fournit un rapport relatif aux appareils qui sont associés à une passerelle. Cela est utile pour établir un mapping d'appareils sans fil à des systèmes hôtes et à des navigateurs de réseau.

Le client interne à une passerelle utilise la primitive G_Device_List_Report pour récupérer un rapport relatif aux appareils associés à une entité de passerelle.

U.2.4.3.2 Primitive G_Device_List_Report

U.2.4.3.2.1 Primitives et leurs paramètres

Le Tableau U.7 décrit l'utilisation des paramètres pour la primitive G_Device_List_Report.

Tableau U.7 – Utilisation des paramètres de la primitive G_Device_List_Report

Nom du paramètre	G_Device_List_Report	
	Demande	Confirmation
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Device_List	-	M
GS_Network_Address	-	M
GS_Device_Type	-	M
GS_Unique_Device_ID	-	M
GS_Manufacturer	-	M
GS_Model	-	M
GS_Revision	-	M
GS_Status	-	M

Le client interne à une passerelle utilise la primitive de demande de G_Device_List_Report pour récupérer un rapport relatif aux appareils associés à une entité de passerelle.

Un identificateur de session (GS_Session_ID) est obtenu à partir de l'interface G_Session et il est inclus dans la demande.

Un identificateur de transaction unique de session (GS_Transaction_ID) est spécifié pour chaque invocation de l'interface.

U.2.4.3.2.2 Utilisation de confirmation de G_Device_List_Report

L'entité de passerelle utilise la primitive de confirmation de G_Device_List_Report pour parachever la demande de G_Device_List_Report au client interne à une passerelle.

L'identificateur de session (GS_Session_ID) et l'identificateur de transaction (GS_Transaction_ID) sont retournés pour permettre l'appariement de la primitive de confirmation avec la primitive de demande d'origine.

Une liste d'appareils associés à l'entité de passerelle (GS_Device_List) est retournée. Pour chaque appareil, la liste inclut l'adresse IPv6 (GS_Network_Address), le type de l'appareil (GS_Device_Type), et l'identificateur d'appareil unique (GS_Unique_Device_ID).

La liste inclut également des informations complémentaires relatives à un fabricant (GS_Manufacturer, GS_Model, et GS_Revision).

Lorsque la passerelle inclut le rôle d'appareil de configuration, l'adresse IPv6 peut être une adresse de défaut. L'identificateur unique et les informations fabricant sont utilisés au sein des applications de niveau hôte pour commander la mise en service de l'appareil par l'intermédiaire de l'interface de configuration d'appareil.

Par exemple, un navigateur peut afficher une liste d'appareils disponibles pour être configurés accompagnée d'informations d'identification. Des appareils d'un ensemble sélectionné sont pris dans l'affichage et sont mis en service avec l'adresse IPv6 et d'autres informations pour rejoindre le système. L'affichage de navigation est alors rafraîchi avec des appareils qui sont disponibles pour la liaison à une stratégie de commande.

GS_Status est retourné pour spécifier le succès ou l'échec de l'opération, conformément au Tableau U.8.

Tableau U.8 – GS_Status pour confirmation de G_Device_List_Report

Valeur	Colonne
0	Succès
1	Echec

U.2.4.4 Interface de rapport de topologie

U.2.4.4.1 Généralités

Dans les configurations de système où l'accès à l'information de gestion de système s'effectue par l'intermédiaire de la passerelle, l'interface de rapport de topologie fournit un rapport de topologie qui relie des appareils au sein d'un maillage sans fil.

Le client interne à une passerelle utilise la primitive G_Topology_Report pour récupérer un rapport relatif aux appareils associés à une entité de passerelle.

U.2.4.4.2 Primitive G_Topology_Report

U.2.4.4.2.1 Primitives et leurs paramètres

Le Tableau U.9 décrit l'utilisation des paramètres pour la primitive G_Topology_Report.

Tableau U.9 – Utilisation des paramètres de la primitive G_Topology_Report

Nom du paramètre	G_Topology_Report	
	Demande	Confirmation
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Device_List	-	M
GS_Network_Address	-	M
GS_Neighbor_List	-	M
GS_Network_Address	-	M
GS_Graph_List	-	M
GS_Graph_ID	-	M
GS_Network_Address	-	M
GS_Status	-	M

U.2.4.4.2.2 Utilisation de demande de G_Topology_Report

Le client interne à une passerelle utilise la primitive de demande de G_Topology_Report pour récupérer un rapport relatif à la topologie des appareils associés à une entité de passerelle.

Un identificateur de session (GS_Session_ID) est obtenu à partir de l'interface G_Session et il est inclus dans la demande.

Un identificateur de transaction unique de session (GS_Transaction_ID) est spécifié pour chaque invocation de l'interface.

U.2.4.4.2.3 Utilisation de confirmation de G_Topology_Report

L'entité de passerelle utilise la primitive de confirmation de G_Topology_Report pour parachever la demande de G_Topology_Report au client interne à une passerelle.

L'identificateur de session (GS_Session_ID) et l'identificateur de transaction (GS_Transaction_ID) sont retournés pour permettre l'appariement d'une primitive de confirmation avec la primitive de demande d'origine.

Une liste d'appareils associés à l'entité de passerelle (GS_Device_List) est retournée. La liste inclut l'adresse IPv6 (GS_Network_Address) pour chaque appareil.

En outre, il est inclus dans la liste une deuxième liste (GS_Neighbor_List) des appareils voisins associés à chaque appareil. La liste inclut l'adresse IPv6 (GS_Network_Address) de tous les voisins (tels que décrits dans le rapport de santé de voisins).

En outre, il est inclus dans la liste une troisième liste (GS_Graph_List) des connexions de graphe associées à chaque appareil. Pour chaque connexion de graphe, la liste inclut le graphe identifié (GS_Graph_ID) et une liste d'adresses IPv6 associée (GS_Network_Address) des voisins sur le graphe.

GS_Status est retourné pour spécifier le succès ou l'échec de l'opération, conformément au Tableau U.8.

U.2.4.5 Interface de rapport de programme

U.2.4.5.1 Généralités

Dans les configurations de système où l'accès à l'information de gestion de système s'effectue par l'intermédiaire de la passerelle, l'interface de rapport de programme fournit un rapport de programme détaillant les allocations d'intervalles de temps et de voies sur une base par appareil.

Le client interne à une passerelle utilise la primitive de demande de G_Schedule_Report pour récupérer un rapport relatif au programme des appareils associés à une entité de passerelle.

U.2.4.5.2 Primitive G_Schedule_Report

U.2.4.5.2.1 Primitives et leurs paramètres

Le Tableau U.10 décrit l'utilisation des paramètres pour la primitive G_Schedule_Report.

Tableau U.10 – Utilisation des paramètres de la primitive G_Schedule_Report

Nom du paramètre	G_Schedule_Report	
	Demande	Confirmation
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Network_Address	M	-
GS_Channel_List	-	M
GS_Channel_Number	-	M
GS_Channel_Status	-	M
GS_Device_Schedule	-	M
GS_Network_Address	-	M
GS_Superframe_List	-	C
GS_Superframe_ID	-	C
GS_Num_Time_Slots	-	C
GS_Start_Time	-	C
GS_Link_List	-	C
GS_Network_Address	-	C
GS_Slot_Size	-	C
GS_Channel	-	C
GS_Direction	-	C
GS_Link_Type	-	C
GS_Status	-	M

U.2.4.5.2.2 Utilisation de demande de G_Schedule_Report

Le client interne à une passerelle utilise la primitive de demande de G_Schedule_List_Report pour récupérer un rapport de programme pour un appareil spécifique associé à une entité de passerelle. L'appareil particulier est identifié par son adresse IPv6 (GS_Network_Address).

Un identificateur de session (GS_Session_ID) est obtenu à partir de l'interface G_Session et il est inclus dans la demande.

Un identificateur de transaction unique de session (GS_Transaction_ID) est spécifié pour chaque invocation de l'interface.

U.2.4.5.2.3 Utilisation de confirmation de G_Schedule_Report

L'entité de passerelle utilise la primitive de confirmation de G_Schedule_Report pour parachever la demande de G_Schedule_Report au client interne à une passerelle.

L'identificateur de session (GS_Session_ID) et l'identificateur de transaction (GS_Transaction_ID) sont retournés pour permettre l'appariement de la primitive de confirmation avec la primitive de demande d'origine.

Une liste de voies est retournée. Chaque élément de la liste inclut le numéro de voie (GS_Channel_Number) et le statut de la voie (GS_Channel_Status). Le statut de la voie est mis à 0 pour indiquer une voie désactivée ou à 1 pour indiquer une voie activée.

Un programme d'appareil (GS_Device_Schedule) est également retourné. Le programme inclut les informations d'identification d'appareil (GS_Network_Address) et une liste de supertrames (GS_Superframe_List) qui sont utilisées par l'appareil pour la communication.

Si un appareil n'utilise pas des supertrames, GS_Superframe_List n'est pas retourné.

La liste des supertrames inclut les informations générales relatives aux supertrames, y compris un identificateur de supertrame (GS_Superframe_ID), le nombre d'intervalles de temps dans la supertrame (GS_Num_Time_Slots), et le temps de début de la supertrame (GS_Start_Time). Seules les supertrames actives sont rapportées.

GS_Start_Time est un décalage par rapport au début du temps TAI. Ce nombre a une signification seulement par rapport au temps de réseau courant, à moins que la communication ne soit synchronisée à une source extérieure. GS_Start_Time est mis à -1 pour indiquer que la supertrame n'a aucune synchronisation connue.

La liste de supertrames inclut également une liste ordonnée (GS_Link_List) avec un élément par intervalle de temps dans la supertrame. Les éléments de liste de liaisons sont utilisés pour décrire des relations de communication relatives à la supertrame. Chaque élément de liste de liaison décrit la durée d'intervalle de temps en microsecondes (GS_Slot_Size) et la voie (GS_Channel) pour la communication au sein de la supertrame. Le paramètre de liaison (GS_Direction) décrit le sens des communications. Une valeur de 0 décrit la réception, et une valeur de 1 décrit l'émission. L'adresse IPv6 (GS_Network_Address) décrit l'adresse IPv6 logique d'un partenaire de communication pour l'intervalle de temps.

Le type de liaison (GS_Link_Type) décrit le but de la communication:

- Une valeur de 0 décrit la communication de données apériodiques.
- Une valeur de 1 décrit la communication de gestion apériodique.
- Une valeur de 2 décrit la communication de données périodiques.
- Une valeur de 3 décrit la communication de gestion périodique.

GS_Status est retourné pour spécifier le succès ou l'échec de l'opération, conformément au Tableau U.8.

U.2.4.6 Interface de rapports de santé d'appareils

U.2.4.6.1 Généralités

L'interface de rapports de santé d'appareil fournit un rapport de santé de communication pour la vue de sa propre santé pour chaque appareil.

Le client interne à une passerelle utilise la primitive G_Device_Health_Report pour récupérer un rapport de santé d'appareil pour un ensemble spécifié d'appareils qui sont associés à une entité de passerelle.

U.2.4.6.2 Primitive G_Device_Health_Report

U.2.4.6.2.1 Primitives et leurs paramètres

Le Tableau U.11 décrit l'utilisation des paramètres pour la primitive G_Device_Health_Report.

Tableau U.11 – Utilisation des paramètres de la primitive G_Device_Health_Report

Nom du paramètre	G_Device_Health_Report	
	Demande	Confirmation
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Device_List	M	M
GS_Network_Address	M	M(=)
GS_DPDUs_Transmitted	-	M
GS_DPDUs_Received	-	M
GS_DPDUs_Failed_Transmission	-	M
GS_DPDUs_Failed_Reception	-	M
GS_Status	-	M

U.2.4.6.2.2 Utilisation de demande de G_Device_Health_Report

Le client interne à une passerelle utilise la primitive de demande de G_Device_Health_Report pour récupérer un rapport de santé d'appareil pour un ensemble spécifié d'appareils qui sont associés à une entité de passerelle.

Un identificateur de session (GS_Session_ID) est obtenu à partir de l'interface G_Session et il est inclus dans la demande.

Un identificateur de transaction unique de session (GS_Transaction_ID) est spécifié pour chaque invocation de l'interface.

Un rapport de santé est demandé pour une liste spécifique d'appareils (GS_Device_List). L'adresse IPv6 de chaque appareil (GS_Network_Address) est exigée.

U.2.4.6.2.3 Utilisation de confirmation de G_Device_Health_Report

L'entité de passerelle utilise la primitive de confirmation de G_Device_Health_Report pour parachever la demande de G_Device_Health_Report au client interne à une passerelle.

L'identificateur de session (GS_Session_ID) et l'identificateur de transaction (GS_Transaction_ID) sont retournés pour permettre l'appariement de la primitive de confirmation avec la primitive de demande d'origine.

Une liste d'appareils (GS_Device_List) est retournée. Elle inclut des informations d'identification d'appareil (GS_Network_Address) et des informations de santé de communication. Les informations de santé de communication incluent le nombre total des DPDU émises (GS_DPDUs_Transmitted) de l'appareil vers tous les voisins, le nombre total des DPDU reçues (GS_DPDUs_Received) de l'appareil par tous les voisins, le nombre total des DPDU vers tous les voisins qui ont échoué à l'émission (GS_DPDUs_Failed_Transmission), ainsi que le nombre total des DPDU provenant de tous les voisins qui ont échoué à la réception (GS_DPDUs_Failed_Reception). Les réceptions échouées incluent les DPDU identifiables qui sont rejetées en raison d'une corruption relative à l'émission.

NOTE Les réceptions échouées seront vraisemblablement moins nombreuses que les émissions échouées, car de nombreuses DPDU échouées n'auront pas assez d'informations non corrompues pour déterminer l'adressage. La réception échouée n'inclut pas des erreurs relatives aux protocoles.

GS_Status est retourné pour spécifier le succès ou l'échec de l'opération, conformément au Tableau U.8.

U.2.4.7 Interface de rapports de santé des voisins

U.2.4.7.1 Généralités

Dans des configurations de système où l'accès à l'information de gestion de système s'effectue par l'intermédiaire de la passerelle, l'interface de rapports de santé des voisins fournit un rapport de santé de communication pour la vue de chaque appareil de ses voisins.

Un appareil voisin est un partenaire de communication sans fil de niveau liaison qui est configuré pour l'échange direct de DPDU (émission RF sans sauts). Les interfaces de rapport de santé de voisins fournissent des informations relatives à ces voisins physiques. Les appareils voisins sont capables de recueillir les statistiques d'échange de DPDU qui indiquent des états RF locales.

Le client interne à une passerelle utilise la primitive G_Neighbor_Health_Report pour récupérer un rapport de santé de communication pour l'ensemble d'appareils voisins qui sont associés à un appareil spécifique qui est associé à une entité de passerelle.

U.2.4.7.2 Primitive G_Neighbor_Health_Report

U.2.4.7.2.1 Primitives et leurs paramètres

Le Tableau U.12 décrit l'utilisation des paramètres pour la primitive G_Neighbor_Health_Report.

Tableau U.12 – Utilisation des paramètres de la primitive G_Neighbor_Health_Report

Nom du paramètre	G_Neighbor_Health_Report	
	Demande	Confirmation
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Network_Address	M	-
GS_Neighbor_Health_List	-	M
GS_Network_Address	-	M
GS_Link_Status	-	M
GS_DPDUs_Transmitted	-	M
GS_DPDUs_Received	-	M
GS_DPDUs_Failed_Transmission	-	M
GS_DPDUs_Failed_Reception	-	M
GS_Signal_Strength	-	M
GS_Signal_Quality	-	M
GS_Status	-	M

U.2.4.7.2.2 Utilisation de demande de G_Neighbor_Health_Report

Le client interne à une passerelle utilise la primitive de demande de G_Neighbor_Health_Report pour récupérer un rapport de santé de communication pour l'ensemble d'appareils voisins qui sont associés à un appareil spécifique qui est associé à une entité de passerelle.

Un identificateur de session (GS_Session_ID) est obtenu à partir de l'interface G_Session et il est inclus dans la demande.

Un identificateur de transaction unique de session (GS_Transaction_ID) est spécifié pour chaque invocation de l'interface.

Un rapport de santé de voisins est demandé pour un appareil en une adresse IPv6 spécifique (GS_Network_Address).

U.2.4.7.2.3 Utilisation de confirmation de G_Neighbor_Health_Report

L'entité de passerelle utilise la primitive de confirmation de G_Neighbor_Health_Report pour parachever la demande de G_Neighbor_Health_Report au client interne à une passerelle.

L'identificateur de session (GS_Session_ID) et l'identificateur de transaction (GS_Transaction_ID) sont retournés pour permettre l'appariement de la primitive de confirmation avec la primitive de demande d'origine.

Une liste de santé de voisins (GS_Neighbor_Health_List) est retournée. La liste inclut des informations d'identification d'appareil voisin (GS_Network_Address) et des informations de santé de communication. Les informations de santé de communication incluent un statut général (GS_Link_Status). GS_Link_Status = 1 indique que le voisin est disponible pour la communication. GS_Link_Status = 0 indique que le voisin est indisponible pour la communication.

Les informations de santé incluent également le nombre des DPDU émises vers le voisin (GS_DPDUs_Transmitted), le nombre des DPDU reçues en provenance du voisin (GS_DPDUs_Received), le nombre de tentatives d'émission échouées (GS_DPDUs_Failed_Transmission), et le nombre de réceptions échouées (GS_DPDUs_Failed_Reception) provenant du voisin. Les réceptions échouées incluent les DPDU identifiables qui sont rejetées en raison d'une corruption relative à l'émission.

NOTE Les réceptions échouées seront vraisemblablement moins nombreuses que les émissions échouées, car de nombreuses DPDU échouées n'auront pas assez d'informations non corrompues pour déterminer l'adressage. La réception échouée n'inclut pas des erreurs relatives aux protocoles.

Les informations de santé incluent également GS_Signal_Strength et GS_Signal_Quality. Ces paramètres retournent des valeurs entre 0 (le plus mauvais signal) et 100 (le meilleur signal). GS_Signal_Strength indique le niveau moyen de puissance non corrélé des signaux reçus en provenance d'un voisin spécifique par rapport à la gamme du récepteur. GS_Signal_Quality indique le niveau moyen de puissance corrélé des signaux reçus en provenance d'un voisin spécifique par rapport à la gamme du récepteur.

GS_Status est retourné pour spécifier le succès ou l'échec de l'opération, conformément au Tableau U.8.

U.2.4.8 Interface de rapports de santé de réseau

U.2.4.8.1 Généralités

Dans des configurations de système où l'accès à l'information de gestion de système s'effectue par l'intermédiaire de la passerelle, l'interface de rapports de santé des voisins fournit un rapport de santé de communication pour la vue de chaque appareil de ses voisins.

Le client interne à une passerelle utilise la primitive G_Network_Health_Report pour récupérer un rapport de santé de communication sommaire pour un réseau tout entier.

U.2.4.8.2 Primitive G_Network_Health_Report

U.2.4.8.2.1 Primitives et leurs paramètres

Le Tableau U.13 décrit l'utilisation des paramètres pour la primitive G_Network_Health_Report.

Tableau U.13 – Utilisation des paramètres de la primitive G_Network_Health_Report

Nom du paramètre	G_Network_Health_Report	
	Demande	Confirmation
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Network_Health	-	M
GS_Network_ID	-	M
GS_Network_Type	-	M
GS_Device_Count	-	M
GS_Start_Date	-	M
GS_Current_Date	-	M
GS_DPDUs_Sent	-	M
GS_DPDUs_Lost	-	M
GS_GPDU_Latency	-	M
GS_GPDU_Path_Reliability	-	M
GS_GPDU_Data_Reliability	-	M
GS_Join_Count	-	M
GS_Device_Health_List	-	M
GS_Network_Address	-	M
GS_Start_Date	-	M
GS_Current_Date	-	M
GS_DPDUs_Sent	-	M
GS_DPDUs_Lost	-	M
GS_GPDU_Latency	-	M
GS_GPDU_Path_Reliability	-	M
GS_GPDU_Data_Reliability	-	M
GS_Join_Count	-	M
GS_Status	-	M

U.2.4.8.2.2 Utilisation de demande de G_Network_Health_Report

Le client interne à une passerelle utilise la primitive de demande de G_Network_Health_Report pour récupérer un rapport de santé de communication sommaire pour un réseau tout entier.

Un identificateur de session (GS_Session_ID) est obtenu à partir de l'interface G_Session et il est inclus dans la demande.

Un identificateur de transaction unique de session (GS_Transaction_ID) est spécifié pour chaque invocation de l'interface.

U.2.4.8.2.3 Utilisation de confirmation de G_Network_Health_Report

L'entité de passerelle utilise la primitive de confirmation de G_Network_Health_Report pour parachever la demande de G_Network_Health_Report au client interne à une passerelle.

L'identificateur de session (GS_Session_ID) et l'identificateur de transaction (GS_Transaction_ID) sont retournés pour permettre l'appariement de la primitive de confirmation avec la primitive de demande d'origine.

Un résumé de santé de réseau (GS_Network_Health) est retourné. Le sommaire inclut des informations d'identification de réseau (GS_Network_ID et GS_Network_Type) et des informations de résumé de santé de communication de réseau. Les informations de santé de communication incluent le nombre d'appareils dans le réseau (GS_Device_Count), la date de début et la date courante pour le réseau (GS_Start_Date et GS_Current_Date), les statistiques d'émission (GS_DPDUs_Sent, GS_DPDUs_Lost, et GS_GPDU_Latency), les statistiques de fiabilité (GS_GPDU_Path_Reliability et GS_GPDU_Data_Reliability), et les statistiques de rattachement (GS_Join_Count).

Un résumé de santé spécifique à un appareil (GS_Device_Health_List) est également retourné. La liste inclut des informations d'identification d'appareil (GS_Network_Address) et les statistiques de communication qui sont un sous-ensemble identique de celles contenues

dans le résumé de santé de réseau (GS_Start_Date, GS_Current_Date, GS_DPDUs_Sent, GS_DPDUs_Lost, GS_GPDU_Latency, GS_GPDU_Path_Reliability, GS_GPDU_Data_Reliability, et GS_Join_Count).

GS_Start_Date est un champ de temps TAI de 48 bits indiquant le moment où un appareil a commencé à fonctionner pour la première fois. Cela est utile pour le calcul des programmes de remplacement de batteries.

GS_Current_Date est un champ de temps TAI de 48 bits indiquant le temps courant tel que vu par l'appareil. Il s'agit du temps utilisé par l'appareil pour les besoins d'horodatage.

GS_GPDU_Latency est un nombre dans la plage 0..100 indiquant le pourcentage des GPDU programmées qui arrivent plus tard que prévu. Ces GPDU peuvent être retardées en raison de la livraison sur des chemins secondaires ou en raison de l'encombrement dans des appareils intermédiaires.

GS_GPDU_Path_Reliability est un nombre dans la plage 0..100 indiquant le pourcentage de succès de premier chemin pour l'émission de GPDU acquittée. Les GPDU qui sont émises sur un chemin secondaire peuvent arriver avec succès, mais elles peuvent réduire la fiabilité du chemin.

GS_GPDU_Data_Reliability est un nombre dans la plage 0..100 indiquant le pourcentage des GPDU totales qui sont des GPDU réussies. Le total des GPDU est le nombre de GPDU d'émission acquittées qui sont tentées plus le nombre des GPDU reçues. Les GPDU réussies sont des GPDU émises acquittées qui sont correctement transférées à la première tentative plus les GPDU de réception qui réussissent aux contrôles d'intégrité.

GS_Join_Count est un nombre entier positif qui indique le nombre de fois qu'un appareil a rejoint le système. Le compte des rattachements peut augmenter si l'alimentation électrique est interrompue, un appareil est réinitialisé, le réseau est reformé, ou un appareil est déplacé vers un nouveau réseau. Des rattachements excessifs peuvent indiquer des problèmes d'intégrité ou de communication d'appareil.

GS_Status est retourné pour spécifier le succès ou l'échec de l'opération, conformément au Tableau U.8.

U.2.4.9 Interface de temps

U.2.4.9.1 Généralités

L'interface de temps permet la récupération et le réglage du temps pour un réseau sans fil associé à une passerelle. Cela est utile pour la synchronisation du temps d'un réseau d'appareils sans fil avec un système hôte et d'autres applications de niveau hôte.

Le client interne à une passerelle utilise la primitive G_Time pour récupérer un rapport relatif aux appareils associés à une entité de passerelle.

U.2.4.9.2 Primitive G_Time

U.2.4.9.2.1 Primitives et leurs paramètres

Le Tableau U.14 décrit l'utilisation des paramètres pour la primitive G_Time.

Tableau U.14 – Utilisation des paramètres de la primitive G_Time

Nom du paramètre	G_Time	
	Demande	Confirmation
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Command	M	-
GS_Time	C	M
GS_Status	-	M

U.2.4.9.2.2 Utilisation de demande de G_Time

Le client interne à une passerelle utilise la primitive de demande de G_Time pour lire ou établir le temps réseau.

Un identificateur de session (GS_Session_ID) est obtenu à partir de l'interface G_Session et il est inclus dans la demande.

Un identificateur de transaction unique de session (GS_Transaction_ID) est spécifié pour chaque invocation de l'interface.

GS_Command = 0 lit le temps réseau. GS_Time n'est pas inclus.

GS_Command = 1 tente de définir le temps réseau. Un nouveau temps (GS_Time) est fourni.

U.2.4.9.2.3 Utilisation de confirmation de G_Time

L'entité de passerelle utilise la primitive de confirmation de G_Time pour parachever la demande de G_Time au client interne à une passerelle.

L'identificateur de session (GS_Session_ID) et l'identificateur de transaction (GS_Transaction_ID) sont retournés pour permettre l'appariement de la primitive de confirmation avec la primitive de demande d'origine.

Si GS_Command = 0 dans la demande, le temps réseau courant (GS_Time) est retourné.

Si GS_Command = 1 dans la demande, l'interface tente d'établir le temps réseau. Le temps réseau courant (GS_Time) est retourné. Si la mise à jour a réussi, le temps courant reflétera la modification.

GS_Status est retourné pour spécifier le succès ou l'échec de l'opération, conformément au Tableau U.15.

Tableau U.15 – GS_Status pour confirmation de G_Time

Valeur	Colonne
0	Succès
1	Echec; non autorisé à définir le temps dans cette configuration
2	Echec; autre

U.2.4.10 Interface client/serveur

U.2.4.10.1 Généralités

L'interface client/serveur permet la prise en charge du transfert de données client/serveur. Les ressources de communication nécessaires pour permettre le transfert sont allouées par l'utilisation de l'interface de location. Le client et le serveur accomplissent, chacun, des rôles distincts, mais connexes. La liaison du client et du serveur est accomplie par l'établissement de locations avec des informations de location concordantes. Les ressources de communication incluent les moyens de tampons locaux afin de réduire au maximum les transactions énergivores. Les clients et les serveurs peuvent exister soit dans la passerelle, soit dans des appareils.

La primitive G_Client_Server est utilisée pour envoyer une charge utile de données de demande client interne à un serveur et pour déclencher la réception d'une charge utile correspondante de données de réponse serveur. En fonction de la mise en œuvre, la charge utile de réponse peut venir en provenance du tampon client interne au sein de la passerelle ou en provenance de l'appareil de terrain.

U.2.4.10.2 Primitive G_Client_Server

U.2.4.10.2.1 Primitives et leurs paramètres

Le Tableau U.16 décrit l'utilisation des paramètres pour la primitive G_Client_Server.

Tableau U.16 – Utilisation des paramètres de la primitive G_Client_Server

Nom du paramètre	G_Client_Server			
	Demande	Affichage	Réponse	Confirmation
GS_Session_ID	M	-	-	M(=)
GS_Transaction_ID	M	-	-	M(=)
GS_Lease_ID	M	-	-	-
GS_Buffer	M	-	-	-
GS_Transfer_Mode	M	-	M	-
GS_Request_Data	M	C(=)	-	-
GS_Response_Data	-	-	C	M
GS_Transaction_Info	C	-	-	C(=)
GS_Status	-	-	M	M

U.2.4.10.2.2 Utilisation de demande de G_Client_Server

La primitive de demande de G_Client_Server est utilisée pour tenter d'acquérir les données demandées localement à partir de la passerelle (si GS_Buffer = 1), pour envoyer une demande correspondante de données d'application client native selon WISN en utilisant le contenu du paramètre de charge utile de données (GS_Request_Data) vers un appareil communiquant selon WISN et pour déclencher la réception d'une réponse serveur selon WISN correspondante. Que les données demandées soient accessibles localement ou à distance, les données sont retournées par l'intermédiaire du paramètre de primitive réponse/confirmation pour la charge utile de réponse (GS_Response_Data).

Un identificateur de session (GS_Session_ID) est obtenu à partir de l'interface G_Session et il est inclus dans la demande.

Un identificateur de transaction unique de session (GS_Transaction_ID) est spécifié pour chaque invocation de l'interface.

L'appareil serveur est connu par l'intermédiaire de l'identificateur de location (GS_Lease_ID) qui a été obtenu à partir de l'interface de location.

Les données de réponse seront demandées à l'appareil serveur si le tampon est désactivé pour la transaction (GS_Buffer = 0). Les données de réponse seront livrées à partir du tampon si le tampon est activé pour la transaction (GS_Buffer = 1) et le tampon contient une réponse concordante qui n'a pas expiré.

GS_Transfer_Mode est fourni avec la demande afin d'indiquer la qualité de l'interface et la priorité pour le transfert des données.

Si GS_Transaction_Info est fourni comme partie intégrante d'une demande, il est retourné par la primitive de confirmation correspondante.

U.2.4.10.2.3 Utilisation d'indication de G_Client_Server

La primitive d'indication de G_Client_Server sert à signaler l'arrivée d'une charge utile de données de demande client au niveau du serveur en vue du traitement.

L'indication est conditionnée à la question de savoir si, oui ou non, la charge utile de données de réponse serveur a pu être livrée à partir du tampon client.

U.2.4.10.2.4 Utilisation de réponse de G_Client_Server

La primitive de réponse de G_Client_Server sert à retourner une charge utile de données de réponse serveur vers le client.

GS_Transfer_Mode est fourni avec la réponse afin d'indiquer la qualité de l'interface et la priorité pour le transfert des données.

La réponse est conditionnée à la question de savoir si, oui ou non, la charge utile de données de réponse serveur a pu être livrée à partir du tampon client.

U.2.4.10.2.5 Utilisation de confirmation de G_Client_Server

La primitive de confirmation de G_Client_Server sert à parachever la demande de G_Client_Server destinée au client.

L'identificateur de session (GS_Session_ID) et l'identificateur de transaction (GS_Transaction_ID) sont retournés pour permettre l'appariement de la primitive de confirmation avec la primitive de demande d'origine.

Une charge utile de données de réponse serveur est retournée. La charge utile est livrée soit à partir du tampon client, soit à partir du serveur.

Si GS_Transaction_Info était fourni dans la demande, il doit être retourné dans la confirmation.

GS_Status est retourné pour spécifier le succès ou l'échec de l'opération, conformément au Tableau U.17.

Tableau U.17 – GS_Status pour confirmation de G_Client_Server

Valeur	Colonne
0	Succès
1	Echec; le serveur n'est pas accessible pour la demande sans mémoire tampon
2	Echec; le serveur n'est pas accessible est la mémoire tampon n'est pas valide pour la demande avec mémoire tampon
3	Echec; la location a expiré
4	Echec; autre

U.2.4.11 Interface publish/subscribe

U.2.4.11.1 Généralités

L'interface publish/subscribe fournit des mécanismes pour le transfert de données publish/subscribe. Les ressources de communication nécessaires pour permettre l'échange de messages sont allouées par l'utilisation de l'interface de location. L'éditeur et l'abonné accomplissent, chacun, des rôles distincts, mais connexes. La liaison de l'éditeur et de l'abonné est accomplie par l'établissement distinct de communication concordante. Les ressources de communication incluent les moyens de tampons locaux dans l'éditeur et l'abonné afin de réduire au maximum les transactions énergivores. Les éditeurs et les abonnés peuvent exister dans les passerelles, dans les adaptateurs, ou dans les appareils natifs.

U.2.4.11.2 Établissement de la location

L'interface G_Lease est utilisée avant l'utilisation de l'interface G_Publish afin d'établir un GS_Lease_ID. Le GS_Lease_Type est mis soit à "publisher", soit à "subscriber", pour configurer le côté respectif et pour établir et réserver les ressources d'entité de passerelle et de voies de communication sous-jacentes.

GS_Network_Address_List est utilisé par l'éditeur et l'abonné pour établir l'identité des autres points d'extrémité. Un éditeur peut décrire plusieurs adresses dans la liste afin de configurer plusieurs abonnés.

Dans les limites de la location, GS_Protocol_Type est utilisé pour décrire le protocole d'application qui sera tunnellisé à travers l'interface. Cela permet que le traitement spécifique à un protocole se produise.

GS_Lease_Parameters est utilisé pour établir l'interaction de protocole attendue entre un éditeur et un abonné.

U.2.4.11.3 Publication

La primitive G_Publish est utilisée par un éditeur pour initier le transfert d'une charge utile de données éditeur vers un ou plusieurs abonnés.

La charge utile de données éditeur est stockée dans un tampon local et transmise du tampon vers les abonnés. Les paramètres de configuration de location déterminent le moment où la transmission sera produira. La transmission se produit afin de respecter les dates limites programmées. Sur une période de temps, la même charge utile peut être transmise plusieurs fois pour indiquer que l'éditeur existe toujours et empêcher la temporisation. Une invocation avec des données inchangées ne peut pas donner lieu à une transmission.

U.2.4.11.4 Abonnement

La primitive G_Subscribe est utilisée par un abonné pour récupérer du tampon local les plus récentes données de publication.

L'abonné reçoit également la plus récente publication associée à une location d'abonné par l'intermédiaire de la primitive d'indication de G_Publish.

La primitive G_Publish_Watchdog est utilisée au sein d'un abonné pour signaler l'expiration d'un temporisateur de chien de garde. Le temporisateur expire en l'absence des mises à jour attendues issues d'un éditeur. Le temporisateur est réinitialisé à l'arrivée de la charge utile de données de publication au niveau de l'abonné. Le temporisateur de chien de garde est configuré comme étant une partie des paramètres de configuration de location.

La primitive G_Publish_Timer est utilisée au sein d'un éditeur pour signaler l'expiration d'un temporisateur de publication. Le temporisateur de publication est un temporisateur périodique qui expire avant la date limite pour transmettre la charge utile de données éditeur. L'indication peut être utilisée pour éditer des données fraîches. Le temporisateur de publication est configuré avec les paramètres de configuration de location.

La primitive G_Subcribe_Timer est utilisée au sein d'un abonné pour signaler l'expiration d'un temporisateur d'abonnement. Le temporisateur d'abonnement est un temporisateur périodique qui expire à la date limite de livraison pour recevoir la charge utile de données éditeur. L'indication est utilisée pour traiter des données de publication existantes. L'arrivée des données fraîches réinitialisera le temporisateur et donnera lieu à une indication G_Publish. Le temporisateur d'abonnement est configuré avec les paramètres de configuration de location.

U.2.4.11.5 Types de primitives et paramètres

U.2.4.11.5.1 Primitive G_Publish et ses paramètres

Le Tableau U.18 décrit l'utilisation des paramètres pour la primitive G_Publish.

Tableau U.18 – Utilisation des paramètres de la primitive G_Publish

Nom du paramètre	G_Publish		
	Demande	Affichage	Confirmation
GS_Session_ID	M	M	M
GS_Transaction_ID	M	-	M(=)
GS_Lease_ID	M	M	-
GS_Transfer_Mode	M	-	-
GS_Publish_Data	M	C(=)	-
GS_Status	-	-	M

U.2.4.11.5.2 Utilisation de demande de G_Publish

La primitive de demande de G_Publish est utilisée pour initier un transfert d'une charge utile de données éditeur (GS_Publish_Data) vers un ou plusieurs abonnés. La charge utile de données éditeur est stockée dans un tampon local et transmise du tampon vers les abonnés.

Un identificateur de session (GS_Session_ID) est obtenu à partir de l'interface G_Session et il est inclus dans la demande.

Un identificateur de transaction unique de session (GS_Transaction_ID) est spécifié pour chaque invocation de l'interface.

L'adressage d'abonné est connu par l'intermédiaire de l'identificateur de location (GS_Lease_ID) qui a été obtenu à partir de l'interface de location.

Dans les paramètres de location, GS_Resource est une valeur identique spécifiée pour un éditeur et un ou plusieurs abonnés afin de faciliter l'établissement de liaison entre les points d'extrémité.

GS_Transfer_Mode est fourni avec la demande afin d'indiquer la qualité de l'interface et la priorité pour le transfert des données.

U.2.4.11.5.3 Utilisation d'indication de G_Publish

La primitive d'indication de G_Publish est utilisée pour signaler l'arrivée d'une charge utile de données éditeur au niveau d'un abonné en vue du traitement.

La charge utile de données éditeur (GS_Publish_Data) est livrée avec l'indication.

L'identificateur de session d'abonné (GS_Session_ID) et l'identificateur de location d'abonné (GS_Lease_ID) sont retournés pour permettre l'association de la primitive d'indication à une relation P/S spécifique.

L'indication est conditionnée à la livraison chronométrée dépendante de la configuration à partir de l'éditeur et indique des données de publication fraîches.

U.2.4.11.5.4 Utilisation de confirmation de G_Publish

La primitive de confirmation de G_Publish sert à parachever la demande de G_Publish.

L'identificateur de session éditeur (GS_Session_ID) et l'identificateur de transaction (GS_Transaction_ID) sont retournés pour permettre l'appariement de la primitive de confirmation avec la primitive de demande d'origine.

GS_Status est retourné pour spécifier le succès ou l'échec de l'opération, conformément au Tableau U.19.

Tableau U.19 – GS_Status pour confirmation de G_Publish

Valeur	Colonne
0	Succès
1	Echec; la location a expiré
2	Echec; autre

U.2.4.11.5.5 Primitive G_Subscribe et ses paramètres

Le Tableau U.20 décrit l'utilisation des paramètres pour la primitive G_Subscribe.

Tableau U.20 – Utilisation des paramètres de la primitive G_Subscribe

Nom du paramètre	G_Subscribe	
	Demande	Confirmation
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Lease_ID	M	-
GS_Publish_Data	-	M
GS_Status	-	M

U.2.4.11.5.6 Utilisation de demande de G_Subscribe

La primitive de demande de G_Subscribe est utilisée pour récupérer du tampon local les plus récentes données de publication (GS_Publish_Data).

Un identificateur de session (GS_Session_ID) est obtenu à partir de l'interface G_Session et il est inclus dans la demande.

Un identificateur de transaction unique de session (GS_Transaction_ID) est spécifié pour chaque invocation de l'interface.

L'adressage d'éditeur est connu par l'intermédiaire de l'identificateur de location (GS_Lease_ID) qui a été obtenu à partir de l'interface de location.

U.2.4.11.5.7 Utilisation de confirmation de G_Subscribe

La primitive de confirmation de G_Subscribe sert à parachever la demande de G_Subscribe.

L'identificateur de session (GS_Session_ID) et l'identificateur de transaction (GS_Transaction_ID) sont retournés pour permettre l'appariement de la primitive de confirmation avec la primitive de demande d'origine.

GS_Status est retourné pour spécifier le succès ou l'échec de l'opération, conformément au Tableau U.21.

Tableau U.21 – GS_Status pour confirmation de G_Subscribe

Valeur	Colonne
0	Réussite; données fraîches
1	Réussite; données périmées
2	Echec; la location a expiré
3	Echec; autre

U.2.4.11.5.8 Primitive G_Publish_Timer et ses paramètres

Le Tableau U.22 décrit l'utilisation des paramètres pour la primitive G_Publish_Timer.

Tableau U.22 – Utilisation des paramètres de la primitive G_Publish_Timer

Nom du paramètre	G_PublishTimer
	Affichage
GS_Session_ID	M
GS_Lease_ID	M

U.2.4.11.5.9 Utilisation d'indication G_Publish_Timer

La primitive d'indication de G_Publish_Timer est utilisée au sein d'un éditeur pour signaler l'expiration d'un temporisateur de publication.

L'identificateur de session publisher (GS_Session_ID) et l'identificateur de location publisher (GS_Lease_ID) sont retournés pour permettre l'association de la primitive d'indication à une relation P/S spécifique.

U.2.4.11.5.10 Primitive G_Subscribe_Timer et ses paramètres

Le Tableau U.23 décrit l'utilisation des paramètres pour la primitive G_Publish_Timer.

Tableau U.23 – Utilisation des paramètres de la primitive G_Subscribe_Timer

Nom du paramètre	G_SubscribeTimer
	Affichage
GS_Session_ID	M
GS_Publish_Data	M
GS_Lease_ID	M

U.2.4.11.5.11 Utilisation d'indication de G_Publish_Timer

La primitive d'indication de G_Subcribe_Timer est utilisée au sein d'un abonné pour signaler l'expiration d'un temporisateur d'abonnement. Le temporisateur est réinitialisé par l'indication de G_Publish.

La charge utile de données éditeur (GS_Publish_Data) est livrée à partir du tampon d'abonné avec l'indication.

L'identificateur de session d'abonné (GS_Session_ID) et l'identificateur de location d'abonné (GS_Lease_ID) sont retournés pour permettre l'association de la primitive d'indication à une relation P/S spécifique.

U.2.4.11.5.12 Primitive G_Publish_Watchdog et ses paramètres

Le Tableau U.24 décrit l'utilisation des paramètres pour la primitive G_Publish_Watchdog.

Tableau U.24 – Utilisation des paramètres de la primitive G_Publish_Watchdog

Nom du paramètre	G_Publish_Watchdog
	Affichage
GS_Session_ID	M
GS_Publish_Data	M
GS_Lease_ID	M

U.2.4.11.5.13 Utilisation d'indication de G_Publish_Watchdog

La primitive d'indication de G_Publish_Watchdog est utilisée au sein d'un abonné pour signaler l'expiration d'un temporisateur de chien de garde en raison de l'absence des mises à jour attendues en provenance d'un éditeur. Le temporisateur est réinitialisé par l'indication de G_Publish.

La charge utile de données éditeur désormais périmées (GS_Publish_Data) est livrée à partir du tampon d'abonné avec l'indication.

L'identificateur de session (GS_Session_ID) et l'identificateur de location (GS_Lease_ID) sont retournés pour permettre l'association de la primitive d'indication à une relation P/S spécifique.

U.2.4.12 Interface de transfert en masse

U.2.4.12.1 Généralités

L'interface de transfert en masse permet la prise en charge du transfert de données en masse. Le transfert de données en masse est utilisé pour transférer des éléments volumineux entre des clients à une passerelle et les appareils sans fil.

Les transferts en masse fonctionnent dans le contexte d'une session entre le fournisseur d'interface de GIAP et l'utilisateur d'interface de GIAP. Toutes les primitives prises en charge par la passerelle par l'intermédiaire d'un GIAP incluent le GS_Session_ID correspondant.

Le client de la session gère les GS_Transaction_IDs propres à une session pour chaque primitive invoquée par le client. Cela est nécessaire afin de maintenir la coordination entre les primitives de transfert en masse.

Le GS_Lease_ID, qui représente les ressources de communication nécessaires allouées au sein de la passerelle, est fourni avec chaque primitive.

Des transferts en masse parallèles distincts sont différenciés par un GS_Transfer_ID. Un GS_Transfer_ID est également inclus dans chaque primitive d'interface de GIAP. L'état de transfert est maintenu pour chaque transfert en masse qui est en cours. Par exemple, le numéro de bloc transféré est maintenu par les points d'extrémité.

G_Bulk_Open sert à ouvrir un transfert en masse. G_Bulk_Close sert à fermer un transfert en masse. G_Bulk_Transfer sert à accomplir le transfert réel de segments de données au sein du transfert en masse.

U.2.4.12.2 Types de primitives et paramètres

U.2.4.12.2.1 Primitive G_Bulk_Open et ses paramètres

Le Tableau U.25 décrit l'utilisation des paramètres pour la primitive G_Bulk_Open.

Tableau U.25 – Utilisation des paramètres de la primitive G_Bulk_Open

Nom du paramètre	G_Bulk_Open	
	Demande	Confirmation
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Lease_ID	M	-
GS_Transfer_ID	M	-
GS_Resource	M	-
GS_Mode	M	-
GS_Block_Size	M	M
GS_Item_Size	C	C
GS_Status	-	M

U.2.4.12.2.2 Utilisation de demande de G_Bulk_Open

La primitive de demande de G_Bulk_Open sert à initier un transfert en masse. L'appareil-cible pour un transfert en masse est sous-entendu par GS_Lease_ID.

L'élément-cible pour un transfert en masse est identifié par GS_Resource.

Un transfert est directionnel (téléchargement montant ou téléchargement descendant) et GS_Mode décrit le sens du transfert. GS_Mode = 0 décrit le téléchargement descendant et GS_Mode = 1 décrit le téléchargement montant.

L'utilisateur d'interface de GIAP positionne GS_Block_Size pour demander une taille de bloc pour la phase de transfert ultérieure.

L'utilisateur d'interface de GIAP positionne le GS_Item_Size pour demander le téléchargement descendant d'un élément d'une taille particulière. L'élément peut dépasser les limites de téléchargement descendant disponibles, ce qui entraîne une réponse d'erreur. GS_Item_Size = 0 demande le téléchargement descendant d'un élément de taille indéterminée.

U.2.4.12.2.3 Utilisation de confirmation de G_Bulk_Open

La primitive de confirmation de G_Bulk_Open est utilisée en réponse à une demande de G_Bulk_Open.

Le GS_Item_Size est positionné par le fournisseur d'interface de GIAP pour indiquer la taille de l'élément. Pour un téléchargement descendant, il s'agit de la taille maximale d'élément qui sera acceptée. Pour un téléchargement montant, il s'agit de la taille réelle d'élément. GS_Item_Size = 0 indique qu'il n'y a aucune limite imposée à la taille d'élément.

Le fournisseur d'interface de GIAP détermine et retourne le GS_Block_Size qui doit être utilisé pour la phase de transfert suivante. La taille de bloc peut être réduite (en fonction des ressources disponibles) par rapport à la taille d'origine demandée dans la demande de GS_Bulk_Open.

GS_Status indique le succès ou l'échec du G_Bulk_Open, conformément au Tableau U.26.

Tableau U.26 – GS_Status pour confirmation de G_Bulk_Open

Valeur	Colonne
0	Succès
1	Echec; l'élément dépasse les limites
2	Echec; ressource inconnue
3	Echec; mode non valide
4	Echec; autre

U.2.4.12.3 Primitive G_Bulk_Transfer et ses paramètres

Le Tableau U.27 décrit l'utilisation des paramètres pour la primitive G_Bulk_Transfer.

Tableau U.27 – Utilisation des paramètres de la primitive G_Bulk_Transfer

Nom du paramètre	G_Bulk_Transfer	
	Demande	Confirmation
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Lease_ID	M	-
GS_Transfer_ID	M	-
GS_Bulk_Data	C	C
GS_Status	-	M

U.2.4.12.3.1 Utilisation de demande de G_Bulk_Transfer

La primitive de demande de G_Bulk_Transfer sert à déplacer des données en masse. GS_Bulk_Data est un segment de transfert qui est conditionnellement envoyé à la cible dans le cas d'un téléchargement.

G_Bulk_Transfer est utilisé autant de fois qu'exigé pour parachever le transfert d'un élément volumineux. G_Bulk_Close est utilisé par l'utilisateur d'interface de GIAP pour indiquer l'achèvement du transfert.

U.2.4.12.3.2 Utilisation de "confirm" G_Bulk_Transfer

La primitive de confirmation de G_Bulk_Transfer est utilisée en réponse à une demande de G_Bulk_Transfer. GS_Bulk_Data est un segment de transfert qui est conditionnellement reçu en provenance de la cible dans le cas d'un chargement.

GS_Status indique le succès ou l'échec du G_Bulk_Transfert, comme indiqué dans le Tableau U.28.

Tableau U.28 – GS_Status pour confirmation de G_Bulk_Transfer

Valeur	Colonne
0	Succès
1	Echec; la communication a échoué
2	Echec; le transfert a été abandonné
3	Echec; autre

U.2.4.12.4 Primitive G_Bulk_Close et ses paramètres

Le Tableau U.29 décrit l'utilisation des paramètres pour la primitive G_Bulk_Close.

Tableau U.29 – Utilisation des paramètres de la primitive G_Bulk_Close

Nom du paramètre	G_Bulk_Close	
	Demande	Confirmation
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Lease_ID	M	-
GS_Transfer_ID	M	-
GS_Status	-	M

U.2.4.12.4.1 Utilisation de demande de G_Bulk_Close

La primitive de demande de G_Bulk_Close est utilisée pour parachever un transfert en masse, et pour nettoyer toute ressource ou tout traitement d'état nécessaire dans le fournisseur d'interface de GIAP.

U.2.4.12.4.2 Utilisation de confirmation de G_Bulk_Close

La primitive de confirmation de G_Bulk_Close est utilisée en réponse à une demande de G_Bulk_Close.

U.2.4.13 Interface d'alerte

U.2.4.13.1 Généralités

L'interface d'alerte permet la prise en charge de l'établissement des événements de notification d'alertes des clients internes à une passerelle. Des opérations complémentaires peuvent être exigées pour recueillir des informations complémentaires relatives à l'alerte ou pour répondre à l'alerte.

Les interfaces d'alerte fonctionnent dans le contexte d'une session entre le fournisseur d'interface de GIAP et l'utilisateur d'interface de GIAP. Toutes les primitives prises en charge par la passerelle par l'intermédiaire d'un GIAP incluent le GS_Session_ID correspondant.

Le client de la session gère les GS_Transaction_IDs en cours qui sont propres à une session pour chaque primitive qu'il invoque. Cela est nécessaire afin de maintenir la coordination entre les primitives d'alertes.

Le GS_Lease_ID, qui représente l'entité de passerelle nécessaire et les ressources de communication nécessaires, est fourni avec chaque primitive.

G_Alert_Subscription est utilisé pour s'abonner à des alertes soit par catégorie, soit par alertes spécifiques.

U.2.4.13.2 Types de primitives et paramètres

U.2.4.13.2.1 Primitive G_Alert_Subscription et ses paramètres

Le Tableau U.30 décrit l'utilisation des paramètres pour la primitive G_Alert_Subscription.

Tableau U.30 – Utilisation des paramètres de la primitive G_Alert_Subscription

Nom du paramètre	G_Alert_Subscription	
	Demande	Confirmation
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Lease_ID	M	-
GS_Subscription_List	M	C
GS_Category	C	C
GS_Network_Address	C	C
GS_Alert_Source_ID	C	C
GS_Subscribe	M	C
GS_Enable	M	C
GS_Status	-	M

U.2.4.13.2.2 Utilisation de demande de G_Alert_Subscription

La primitive de demande G_Alert_Subscription sert à gérer une liste d'abonnements d'alertes.

GS_Subscription_List contient une ou plusieurs demandes de modification d'abonnements d'alertes. Chaque élément de liste peut être utilisé pour modifier l'abonnement pour une catégorie particulière d'alertes ou pour modifier l'abonnement pour une alerte spécifique issue d'une source spécifique.

Les éléments de liste peuvent décrire le paramètre GS_Category pour indiquer la modification d'abonnement pour une catégorie particulière d'alertes. Les catégories d'alertes comprennent:

0 = appareil;

1 = réseau;

2 = sécurité; et

3 = processus.

GS_Network_Address et GS_Alert_Source_ID ne sont pas fournis si GS_Category est fourni.

En variante, les éléments de liste peuvent décrire GS_Network_Address et GS_Alert_Source_ID au lieu de GS_Category pour décrire un appareil spécifique et un identificateur d'une alerte spécifique issue de l'appareil en question.

NOTE Il est prévu que certains clients internes à une passerelle, tels que des systèmes complets de gestion d'alarme, utiliseront des catégories complètes, tandis que d'autres clients internes à une passerelle sont capables de restreindre leur utilisation au seul ensemble sélectionné d'alertes.

GS_Subscribe et GS_Enable commandent les actions pour chaque élément de liste. GS_Subscribe sert à décrire quelles alertes doivent être reçues au sein de l'entité de passerelle et être transmises vers l'utilisateur d'interface de GIAP sous la forme d'indications de G_Alert_Notification. GS_Enable sert à commander la production sous-jacente d'alertes au niveau de la source. GS_Subscribe = 1 abonne à une alerte spécifique ou à une catégorie d'alertes, alors que GS_Subscribe = 0 désabonne de l'alerte. GS_Enable = 1 active une alerte spécifique ou une catégorie d'alertes, alors que GS_Enable = 0 désactive l'alerte.

Afin de synchroniser l'état d'alarme entre la source d'alarme et l'entité de passerelle, la récupération d'alarme est initiée sur des abonnements.

U.2.4.13.2.3 Utilisation de confirmation de G_Alert_Subscription

La primitive de confirmation de G_Alert_Subscription est utilisée en réponse à une demande de G_Alert_Subscription.

GS_Status indique le succès ou l'échec de la demande de G_Alert_Subscription, comme indiqué dans le Tableau U.31. Un paramètre GS_Subscription_List avec un seul élément est retourné conditionnellement si l'opération échoue. Le code statut se rapporte à l'élément particulier. Le traitement de la liste s'arrête au premier élément défaillant.

Tableau U.31 – GS_Status pour confirmation de G_Alert_Subscription

Valeur	Colonne
0	Succès
1	Echec; catégorie non valide
2	Echec; alerte individuelle non valide
3	Echec; autre

U.2.4.13.2.4 Primitive G_Alert_Notification et ses paramètres

Le Tableau U.32 décrit l'utilisation des paramètres pour la primitive G_Alert_Notification.

Tableau U.32 – Utilisation des paramètres de la primitive G_Alert_Notification

Nom du paramètre	G_Alert_Notification
	Affichage
GS_Session_ID	M
GS_Lease_ID	M
GS_Alert	M
GS_Network_Address	M
GS_Alert_Source_ID	M
GS_Time	M
GS_Class	M
GS_Direction	M
GS_Category	M
GS_Type	M
GS_Priority	M
GS_Alert_Data	C

U.2.4.13.2.5 Utilisation de confirmation de G_Alert_Notification

L'indication de G_Alert_Notification est générée par le fournisseur d'interface de GIAP et envoyée à l'utilisateur de GIAP en réponse à une alerte reçue par la passerelle. Une notification est fournie seulement pour les alertes auxquelles le client de GIAP s'est abonné et pour lesquelles la notification a été activée.

Une structure GS_Alert est fournie dans l'indication pour fournir les détails spécifiques d'alertes comme suit:

- GS_Network_Address indique l'appareil source de l'alerte.
- GS_Alert_Source_ID indique l'alerte spécifique au sein de l'appareil source.

- GS_Time est un marqueur temporel qui indique le moment auquel l'alerte a été produite à l'origine.
- GS_Class = 0 identifie l'alerte comme étant un événement; GS_Class = 1 identifie l'alerte comme étant une alarme.
- GS_Direction classe en plus les alarmes comme suit:
  0: la condition d'alarme est terminée;
  1: la condition d'alarme a commencé.
- GS_Category décrit la catégorie alertes comme suit:
  0: relatif au processus;
  1: relatif à l'appareil;
  2: relatif au réseau;
  3: relatif à la sécurité.
- GS_Type décrit les sous-catégories pour les alertes. La valeur réelle est spécifique à une application.
- GS_Priority décrit une priorité pour l'alerte. Plus la valeur est élevée, plus la priorité est haute. La valeur réelle est spécifique à une application.
- GS_Alert_Data autorise l'inclusion d'informations relatives à une alerte. Ce champ est conditionné à la disponibilité ou à la non-disponibilité d'informations d'alerte complémentaires. La valeur réelle est spécifique à une application.

L'entité de passerelle acquitte la réception de l'alerte.

U.2.4.14 Interface de configuration de passerelle

U.2.4.14.1 Généralités

L'interface de configuration de passerelle permet la prise en charge de la lecture et de l'écriture des attributs de configuration de la passerelle.

Le client interne à une passerelle utilise la primitive G_Read_Gateway_Configuration pour récupérer des attributs de configuration de passerelle.

U.2.4.14.2 Types de primitives et paramètres

U.2.4.14.2.1 Primitive G_Read_Gateway_Configuration et ses paramètres

Le Tableau U.33 décrit l'utilisation des paramètres pour la primitive G_Read_Gateway_Configuration.

Tableau U.33 – Utilisation des paramètres de la primitive G_Read_Gateway_Configuration

Nom du paramètre	G_ReadGatewayConfiguration	
	Demande	Confirmation
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Attribute_Identifier	M	-
GS_Attribute_Value	-	C
GS_Status	-	M

U.2.4.14.2.2 Utilisation de demande de G_Read_Gateway_Configuration

Le client interne à une passerelle utilise la primitive de demande de G_Read_Gateway_Configuration pour récupérer des paramètres de configuration de passerelle.

Un identificateur de session (GS_Session_ID) est obtenu à partir de l'interface G_Session et il est inclus dans la demande.

Un identificateur de transaction unique de session (GS_Transaction_ID) est spécifié pour chaque invocation de l'interface.

L'attribut demandé est spécifié par l'identificateur d'attribut (GS_Attribute_Identifier), conformément au Tableau 34. La valeur demandée est spécifiée par la valeur d'attribut (GS_Attribute_Value).

Tableau 34 – Valeurs de GS_Attribute_Identifier pour la demande de G_Read_Gateway_Configuration

Valeur	Colonne
0	GS_GUID
1	GS_Max_Retries
2	GS_Max_Devices
3	GS_Actual_Devices

U.2.4.14.2.3 Utilisation de confirmation de G_Read_Gateway_Configuration

L'entité de passerelle utilise la primitive de confirmation de G_Read_Gateway_Configuration pour parachever la demande de G_Read_Gateway_Configuration au client interne à une passerelle.

L'identificateur de session (GS_Session_ID) et l'identificateur de transaction (GS_Transaction_ID) sont retournés pour permettre l'appariement de la primitive de confirmation avec la primitive de demande d'origine.

Si l'opération réussit, la valeur (GS_Attribute_Value) est retournée pour l'attribut demandé (GS_Attribute_Identifier).

GS_Status est retourné pour spécifier le succès ou l'échec de l'opération, conformément au Tableau U.8.

U.2.4.14.2.4 Primitive G_Write_Gateway_Configuration et ses paramètres

Le Tableau U.35 décrit l'utilisation des paramètres pour la primitive G_Write_Gateway_Configuration.

Tableau U.35 – Utilisation des paramètres de la primitive G_Write_Gateway_Configuration

Nom du paramètre	G_Write_Gateway_Configuration	
	Demande	Confirmation
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Attribute_Identifier	M	-
GS_Attribute_Value	M	-
GS_Status	-	M

U.2.4.14.2.5 Utilisation de demande de G_Write_Gateway_Configuration

Le client interne à une passerelle utilise la primitive de demande de G_Write_Gateway_Configuration pour modifier des attributs de configuration de passerelle.

Un identificateur de session (GS_Session_ID) est obtenu à partir de l'interface G_Session et il est inclus dans la demande.

Un identificateur de transaction unique de session (GS_Transaction_ID) est spécifié pour chaque invocation de l'interface.

L'attribut demandé est spécifié par l'identificateur d'attribut (GS_Attribute_Identifier), conformément au Tableau U.36. La valeur demandée est spécifiée par la valeur d'attribut (GS_Attribute_Value).

Tableau U.36 – Valeurs de GS_Attribute_Identifier pour la demande de G_Write_Gateway_Configuration

Valeur	Colonne
0	GS_GUID
1	GS_Max_Retries

U.2.4.14.2.6 Utilisation de confirmation de G_Write_Gateway_Configuration

L'entité de passerelle utilise la primitive de confirmation de G_Write_Gateway_Configuration pour parachever la demande de G_Write_Gateway_Configuration au client interne à une passerelle.

L'identificateur de session (GS_Session_ID) et l'identificateur de transaction (GS_Transaction_ID) sont retournés pour permettre l'appariement de la primitive de confirmation avec la primitive de demande d'origine.

GS_Status est retourné pour spécifier le succès ou l'échec de l'opération, conformément au Tableau U.37.

Tableau U.37 – GS_Status pour confirmation de G_Write_Gateway_Configuration

Valeur	Colonne
0	Succès
1	Echec; valeur d'attribut non valide
2	Echec; autre

U.2.4.15 Interface de configuration d'appareil

U.2.4.15.1 Généralités

L'interface de configuration d'appareil fournit une méthode pour gérer la configuration des appareils qui sont associés à une passerelle. Cela est utile pour la mise en service d'appareils sans fil pour des systèmes hôtes et des applications connexes.

La configuration d'appareil a des interfaces pour réécrire et pour relire la configuration pour un ou plusieurs appareils.

Un identificateur unique est utilisé pour apparier la configuration à un appareil spécifique. Une adresse IPv6 est spécifiée pour être utilisée dans la configuration de l'appareil, permettant l'accès logique ultérieur de l'appareil.

L'interface de rapport de liste d'appareils est utilisée pour déterminer les appareils associés à la passerelle. Cette interface fonctionne conjointement avec l'interface de rapports de listes d'appareils en fournissant la capacité de limiter les appareils qui sont associés à une passerelle.

Un fichier de configuration peut être fourni pour chaque appareil. Le format d'un tel fichier de configuration est dépendant d'une mise en œuvre de passerelle. Les informations contenues dans ce fichier visent à permettre à des passerelles de configurer automatiquement des appareils pour rejoindre le réseau. Une configuration supplémentaire est accomplie de façon biunivoque par les interfaces client serveur et de transfert en masse.

Si la passerelle est utilisée pour configurer des appareils, le rapport de listes d'appareil doit être vide jusqu'à ce que les appareils soient configurés et rejoignent le système.

Le client interne à une passerelle utilise la primitive G_Write_Device_Configuration pour établir la configuration pour des appareils associés à une entité de passerelle.

U.2.4.15.2 Types de primitives et paramètres

U.2.4.15.2.1 Primitive G_Write_Device_Configuration et ses paramètres

Le Tableau U.38 décrit l'utilisation des paramètres pour la primitive G_Write_Device_Configuration.

Tableau U.38 – Utilisation des paramètres de la primitive G_Write_Device_Configuration

Nom du paramètre	G_Write_Device_Configuration	
	Demande	Confirmation
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Device_List	M	-
GS_Configure	M	-
GS_Unique_Device_ID	M	-
GS_Network_Address	M	-
GS_Provisioning_Info	U	-
GS_Status	-	M

U.2.4.15.2.2 Utilisation de demande de G_Write_Device_Configuration

Le client interne à une passerelle utilise la primitive de demande de G_Write_Device_Configuration pour configurer les appareils associés à une entité de passerelle.

Un identificateur de session (GS_Session_ID) est obtenu à partir de l'interface G_Session et il est inclus dans la demande.

Un identificateur de transaction unique de session (GS_Transaction_ID) est spécifié pour chaque invocation de l'interface.

Une liste d'appareils associés à l'entité de passerelle (GS_Device_List) est fournie. Pour chaque appareil dans la liste, l'identificateur unique d'appareil (GS_Unique_Device_ID) indique l'appareil associé à la configuration. Si GS_Configure = 1, la configuration est ajoutée pour l'appareil spécifique, tandis que si GS_Configure = 0, la configuration est retirée pour l'appareil spécifique. Une adresse IPv6 (GS_Network_Address) concordante indique l'adresse logique à associer à l'appareil.

Les informations de configuration d'appareil (GS_Provisioning_Info) sont fournies à la passerelle pour que la passerelle commande la configuration de l'appareil.

U.2.4.15.2.3 Utilisation de confirmation de G_Write_Device_Configuration

L'entité de passerelle utilise la primitive de confirmation de G_Write_Device_Configuration pour parachever la demande de G_Write_Device_Configuration au client interne à une passerelle.

L'identificateur de session (GS_Session_ID) et l'identificateur de transaction (GS_Transaction_ID) sont retournés pour permettre l'appariement de la primitive de confirmation avec la primitive de demande d'origine.

GS_Status est retourné pour spécifier le succès ou l'échec de l'opération, conformément au Tableau U.39.

Tableau U.39 – GS_Status pour confirmation de G_Write_Device_Configuration

Valeur	Colonne
0	Succès
1	Echec; adresse IPv6 non valide ou en double
2	Echec; mémoire saturée
3	Echec; appareils maximaux de la passerelle dépassés
4	Echec; informations de configuration non valide
5	Echec; autre

U.2.4.15.2.4 Primitive G_Read_Device_Configuration et ses paramètres

Le Tableau U.40 décrit l'utilisation des paramètres pour la primitive G_Read_Device_Configuration.

Tableau U.40 – Utilisation des paramètres de la primitive G_Read_Device_Configuration

Nom du paramètre	G_Read_Device_Configuration	
	Demande	Confirmation
GS_Session_ID	M	M(=)
GS_Transaction_ID	M	M(=)
GS_Device_List	U	M
GS_Unique_Device_ID	U	M
GS_Network_Address	-	M
GS_Provisioning_Info	-	U
GS_Status	-	M

U.2.4.15.2.5 Utilisation de demande de G_Read_Device_Configuration

Le client interne à une passerelle utilise la primitive de demande de G_Read_Device_Configuration pour récupérer la configuration des appareils associés à une entité de passerelle.

Un identificateur de session (GS_Session_ID) est obtenu à partir de l'interface G_Session et il est inclus dans la demande.

Un identificateur de transaction unique de session (GS_Transaction_ID) est spécifié pour chaque invocation de l'interface.

Si la liste d'appareils associés à l'entité de passerelle (GS_Device_List) est fournie, la demande s'applique à ces appareils spécifiques et l'identificateur unique d'appareil (GS_Unique_Device_ID) indique l'appareil associé à la configuration devant être lue. Si aucune liste n'est fournie, la demande s'applique à tous les appareils.

U.2.4.15.2.6 Utilisation de confirmation de G_Read_Device_Configuration

L'entité de passerelle utilise la primitive de confirmation de G_Read_Device_Configuration pour parachever la demande de G_Read_Device_Configuration au client interne à une passerelle.

L'identificateur de session (GS_Session_ID) et l'identificateur de transaction (GS_Transaction_ID) sont retournés pour permettre l'appariement de la primitive de confirmation avec la primitive de demande d'origine.

Une liste d'appareils associés à l'entité de passerelle (GS_Device_List) est retournée. Pour chaque appareil dans la liste, l'identificateur unique d'appareil (GS_Unique_Device_ID) indique l'appareil associé à la configuration. Une adresse IPv6 (GS_Network_Address) concordante indique l'adresse logique associée à l'appareil. Le fichier de configuration d'appareil (GS_Provisioning_Info), lorsqu'il est présent, fournit des informations de configuration pour l'appareil.

GS_Status est retourné pour spécifier le succès ou l'échec de l'opération, conformément au Tableau U.8.

U.3 Utilisations exemplaires de services et objets normalisés WISN

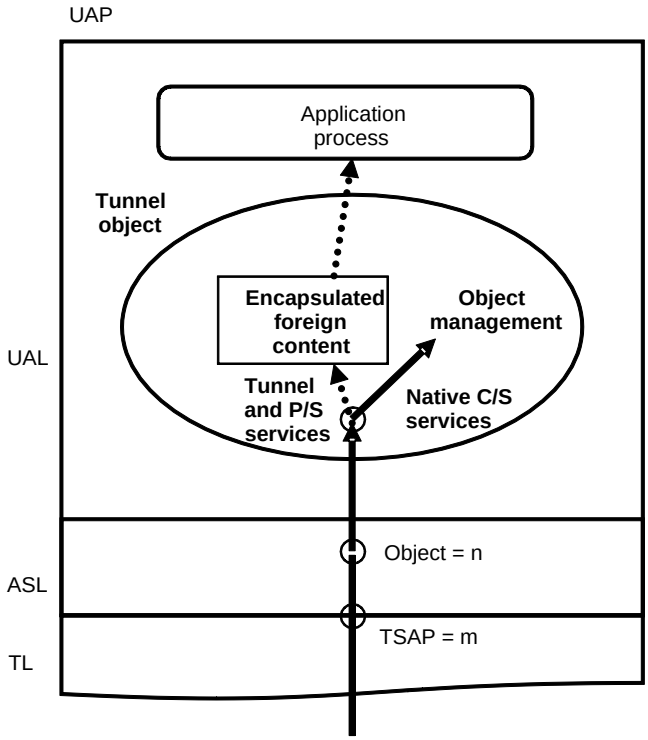
U.3.1 Tunnellisation

U.3.1.1 Généralités

L'objet tunnel (TUN) est un objet natif qui agit comme un point d'extrémité de communication pour la messagerie suivante:

- contenu de protocole étranger encapsulé (représenté sous la forme d'une ligne pointillée à la Figure U.16); et
- contenu d'interface natif (montré sous la forme d'une ligne en trait continu à la Figure U.16) pour configurer et gérer l'objet tunnel.

Les processus de passerelles et les processus d'adaptateurs utilisent des objets tunnel pour prendre en charge la conversion de protocoles étrangers. Un aspect important de l'objet TUN est qu'il fournit le comportement de message tamponné pour le contenu étranger.



Légende

Anglais	Français
Application process	Processus d'application
Tunnel object	Objet tunnel
Encapsulated foreign content	Contenu étranger encapsulé
Object management	Gestion d'objet
Tunnel and P/S services	Services tunnel et P/S
Native C/S services	Services C/S natifs
Object = n	Objet = n
TSP = m	TSP = m

Figure U.16 – Modèle d'objet tunnel

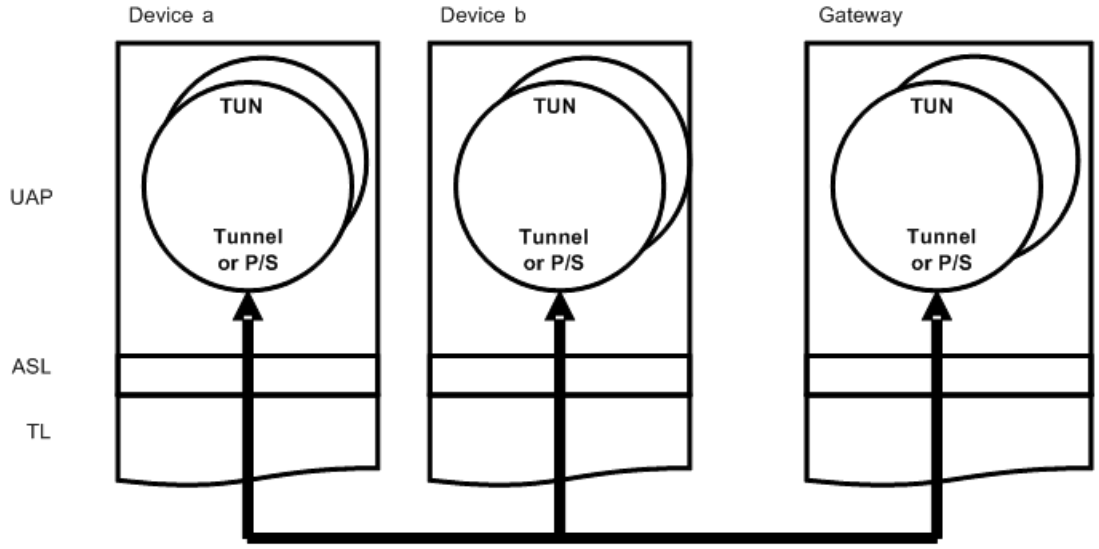
Un ou plusieurs TUN peuvent exister au sein d'un UAP.

Chaque objet TUN peut traiter un protocole étranger complet ou une partie de celui-ci. Les appareils qui traitent plusieurs protocoles étrangers auront besoin de mettre en œuvre plusieurs TUN. L'objet TUN est indépendant vis-à-vis du protocole étranger.

L'objet TUN s'appuie sur la sous-couche d'application (ASL) afin d'acheminer des messages entre les TUN homologues et entre un objet TUN et d'autres objets autres que TUN.

U.3.1.2 Distribution des objets tunnel

Chaque appareil peut avoir un ou plusieurs TUN. Les appareils de tunnellisation comportent au moins un objet TUN comme point d'extrémité pour des tunnels. La Figure U.17 montre un groupe d'appareils connexes avec des points d'extrémité de tunnel interconnectés entre TUN.



Légende

Anglais	Français
Device-a	Appareil a
Device-b	Appareil b
Gateway	Passerelle
Tunnel or P/S	Tunnel ou P/S

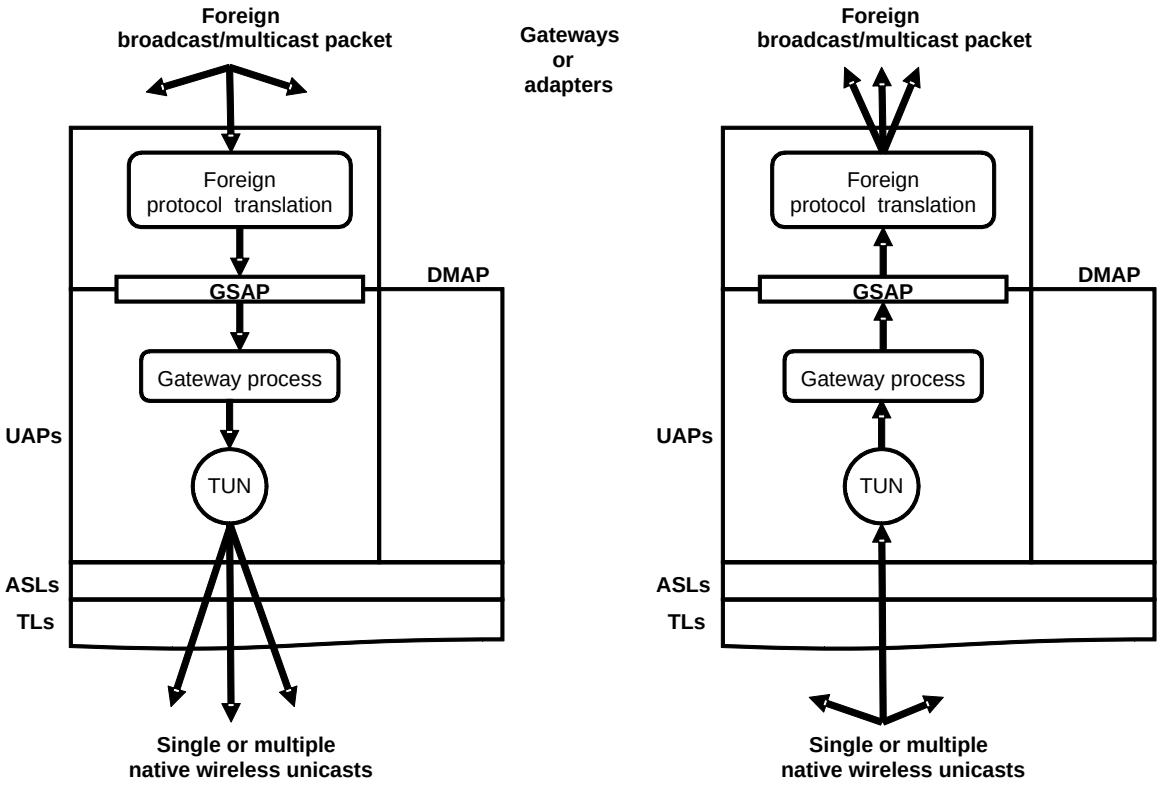
Figure U.17 – Points d'extrémité de tunnels distribués

Les appareils de terrain et les adaptateurs peuvent contenir des TUN qui coopèrent avec d'autres TUN dans une passerelle. Un groupe de TUN connexes communique par l'intermédiaire d'un protocole étranger commun. L'utilisation typique des TUN est de communiquer entre les appareils d'extrémité et un système hôte par l'intermédiaire de la passerelle. Le tunnellisation direct appareil à appareil est également prise en charge au sein du modèle d'objet.

La communication d'objet TUN est établie en utilisant des interfaces TL invoquées et augmentées par l'intermédiaire de l'ASL. Les relations de communication comportent l'édition, l'abonnement, le tunnel à deux parties et le tunnel à quatre parties. Plusieurs relations peuvent être établies simultanément.

U.3.1.3 Messagerie en multidiffusion, en diffusion et "un à plusieurs"

Comme montré à la Figure U.18, les protocoles étrangers peuvent exiger la conversion de relations de messagerie en diffusion/multidiffusion lors de l'utilisation d'interfaces telles que P/S et la distribution d'alertes. Cette messagerie requiert la prise en charge de la conversion au sein de la présente norme.



Légende

Anglais	Français
Foreign broadcast/multicast packet	Paquet étranger en diffusion/multidiffusion
Foreign protocol translation	Conversion de protocole étranger
Gateway process	Processus de passerelle
Single or multiple native wireless unicasts	Une seule ou plusieurs monodiffusions sans fil natives
Gateways or adapters	Passerelles ou adaptateurs

Figure U.18 – Messagerie en multidiffusion, en diffusion et "un à plusieurs"

La présente norme fournit la prise en charge de la messagerie "un à plusieurs" dans l'objet tunnel afin de prendre en charge les demandes en monodiffusion et en diffusion de conversion du protocole étranger. Les couches sous-jacentes de la suite de protocoles ne fournissent pas d'interfaces de diffusion ou de multidiffusion à l'AL. La messagerie "un à plusieurs" est réalisée par l'intermédiaire d'une série d'opérations en monodiffusion. Les applications de conversion de protocoles ne peuvent pas s'appuyer sur une livraison simultanée de messages en monodiffusion.

U.3.1.4 Comportement de messages tamponnés tunnel

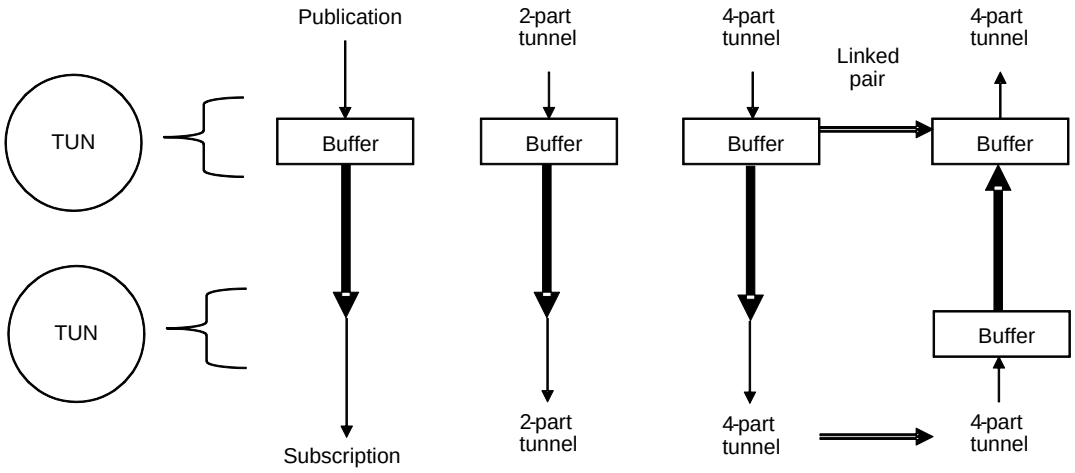
La communication d'objet TUN peut être mise en œuvre pour fournir des capacités pour le comportement tamponné et non tamponné pour des échanges de messages P/S et basés sur des tunnels. Les TUN sont capables de traiter de manière coopérative le comportement tamponné pour réduire les transactions sans fil.

NOTE 1 Certains protocoles hérités utilisent les échanges de messagerie tamponnés pour prendre en charge une conversion de protocoles à haut rendement énergétique et haute performance.

NOTE 2 Certaines applications sont incapables de tolérer le comportement tamponné, habituellement en raison d'exigences relatives à la sécurité et à la synchronisation.

NOTE 3 Les tampons ont une profondeur d'un seul élément. Rien dans la présente norme n'empêche la mise en œuvre d'améliorations de la mise en cache et du placement en file d'attente.

Comme montré à la Figure U.19, chaque point d'extrémité d'un flux de communication a une différente responsabilité relative au placement en tampon, selon la relation.



Légende

Anglais	Français
Publication	Publication
2-part tunnel	Tunnel à deux parties
4-part tunnel	Tunnel à quatre parties
Linked pair	Paire reliée
Buffer	Tampon
Subscription	Abonnement

Figure U.19 – Placement en tampon des objets tunnel

La Figure U.19 montre deux objets TUN. Les flèches en trait fin indiquent des interactions issues d'applications de tunnel qui utilisent les objets. Les flèches en trait épais indiquent les flux de messages à travers le réseau entre les objets. Trois types de transaction sont montrées, une transaction P/S, une transaction de tunnel à deux parties et une transaction de tunnel à quatre parties. Des tampons sont montrés pour illustrer le comportement de messagerie tamponné entre les points d'extrémité de tunnel.

Un éditeur et un abonné sont reliés d'un objet TUN à un objet TUN pour les mises à jour périodiques. Les éditeurs utilisent des publications de tampons de changement d'état pour d'éviter d'envoyer des informations répétées; les abonnés tolèrent des intervalles limités avec des publications absentes.

La messagerie avec l'interface de tunnel à deux parties se comporte d'une façon similaire, excepté que la messagerie est apériodique.

La messagerie avec l'interface de tunnel à quatre parties distingue un côté demande et un côté réponse par l'intermédiaire du bit de demande/ réponse dans l'en-tête de l'interface de tunnel. Le côté demande met en tampon la première demande et transmet également la demande. Une réponse est générée telle qu'indiqué par la double flèche sur le côté réponse à la Figure U.19. La réponse est stockée dans un deuxième tampon lié du côté demande, comme indiqué par la double flèche à la Figure U.19. Le traitement à changement d'état s'applique aux demandes en doublons ultérieures, dans lesquelles la réponse est retournée en provenance du tampon local. Lorsque le changement d'état indique une demande altérée, la demande est transmise et le tampon local de réponse est mis à jour. Un tampon final est montré à la Figure U.19 sur le côté réponse. Ce tampon prend en charge le comportement de changement d'état dans lequel une seule demande produit plusieurs réponses au fil du temps pour mettre à jour le côté demande.

U.3.1.5 Attributs d'objet tunnel

Les attributs d'un objet TUN sont décrits à l'Article 12, mais font l'objet d'une description plus approfondie dans le présent document.

L'attribut Protocol est utilisé pour configurer le protocole associé à l'objet tunnel et aux objets tunnel distants associés. Lorsque le protocole est mis à none (aucun), le tunnel peut être configuré. Une fois qu'un autre protocole est positionné, la configuration d'objet tunnel est appliquée et le statut est mis à jour pour refléter le résultat.

La structure de points d'extrémité de tunnel décrit les informations d'adresses pointant vers un seul objet tunnel distant. La matrice de points d'extrémité de tunnel permet la spécification d'un ou plusieurs points d'extrémité de tunnel représentant des objets tunnel distants. Cela permet à une seule relation de communication de couvrir plusieurs objets tunnel si besoin est. Max_Peer_tunnels indique le nombre maximal d'entrées dans la matrice. Num_Peer_tunnels indique le nombre réel d'entrées configurées dans la matrice.

L'un de plusieurs types de types de flux de communication est choisi entre les objets tunnel par la configuration de l'attribut de Flow_Type. Les types de flux comprennent le tunnel en deux parties, le tunnel à quatre parties, "publish" et "subscribe".

Pour les Flow_Types "publish" et "subscribe", l'Update_Policy permet la configuration de la publication périodique ou de la publication de changement d'état. La publication périodique se produit à chaque occasion. La publication CoSt se produit seulement lorsque des données de publication fraîches sont disponibles. La fréquence de publication est fonction de l'attribut Period. La temporisation réelle est basée sur une combinaison des attribut Period et Phase. Le Stale_Limit est utilisé dans l'abonné pour configurer le comportement pour la détection de la perte excessive ou du retard excessif de publications. Stale_Limit est un multiplicateur qui configure le nombre de périodes qu'un abonné attendra avant de considérer des publications perdues pour indiquer un problème.

Foreign_Destination_Address et Foreign_Source_Address sont les adresses associées au point d'extrémité de tunnel par le protocole étranger. Le format est fonction du protocole étranger. Ces adresses sont retournées aux applications de conversion de protocole à mesure que les messages d'objet tunnel sont reçus. Elles permettent l'utilisation de l'adressage IPv6 tel que défini dans la présente norme au lieu de transporter l'adressage étranger. Le mapping par l'intermédiaire de l'objet tunnel permet la reconstruction des PDU étrangères contenant des informations relatives aux adresses.

NOTE La PDU étrangère variera en fonction de la conversion de protocole étranger spécifique. La plupart des protocoles de bus de terrain formeront des DPDU pour une livraison directe sur une liaison locale. Par contraste, les protocoles basés sur IP forment habituellement des NPDU, où une encapsulation finale est réalisée par un protocole de résolution d'adresse.

Connection_Info[ ] et Transaction_Info[ ] sont des chaînes d'octets qui sont écrites par le convertisseur de protocole, le cas échéant. Connection_Info[ ] sert à fournir le contenu de message statique spécifique à un protocole à la réception de message afin d'éliminer le transfert de message sans fil répété du contenu. Transaction_Info[ ] sert à fournir le contenu de message spécifique à un protocole à la réception d'une réponse, où le contenu aurait autrement été repris en écho à partir de la demande dans la réponse, éliminant le transfert sans fil du contenu. Une description supplémentaire est fournie dans U.3.1.9 et Annexe O.

Il est de la responsabilité de la mise en œuvre de l'objet TUN de maintenir un contrat connexe pour chaque point d'extrémité de tunnel.

U.3.1.6 Messagerie d'objet tunnel

U.3.1.6.1 Utilisation d'interface de sous-couche d'application

Des objets TUN peuvent être mis en œuvre pour fournir les interfaces de connexion qui comprennent une interface P/S, une interface de tunnel à deux parties de tunnel, et une

interface de tunnel à quatre parties de tunnel. Chaque interface peut être mise en œuvre dans un mode de fonctionnement tant tamponné que non tamponné.

Une interface externe facultative pour invoquer les interfaces de connexion de passerelles est décrite à l'Article U.2.

L'objet TUN utilise l'ASL pour livrer et recevoir le contenu d'interface tel que décrit pour l'interface "publish" en 12.17.3.2 et pour l'interface tunnel en 12.17.6.2. L'ASL fournit la livraison objet vers objet de charges utiles P/S dans des formats externes par l'intermédiaire de la primitive de demande de d'éditeur. L'ASL fournit également une primitive de demande de tunnel et une primitive de réponse de tunnel qui sont liées.

L'en-tête est décrit en 12.22.2.3. Cet en-tête permet la spécification des demandes et des réponses, la spécification du type d'interface ("publish" ou tunnel), et le mode d'adressage d'identificateur d'objet (de 4 bits, de 8 bits ou de 16 bits). Un grand nombre d'objets tunnel conduira à un espace adresse plus grand et à plus de surdébit dans l'en-tête.

Le format de charge utile d'interface "publish" est décrit en 12.22.2.12 par le Tableau 348. Il n'y a aucune taille explicite dans l'en-tête. La taille de la publication est fournie avec la demande d'édition et est connue de l'abonné grâce aux informations fournies avec l'indication.

Les formats de charge utile de demande et de réponse d'interface tunnel sont décrits en 12.22.2.9. La demande autorise une taille de 7 bits (charges utiles de 0..127 octets) ou une taille de 15 bits (charges utiles de 128..32 767 octets). L'inclusion de la taille permet au message de tunnel d'être concaténé par l'ASL.

NOTE La plupart des messages encapsulés provenant de protocoles hérités référencés par la présente norme s'inscrivent dans la plage inférieure à une charge utile de 127 octets, conduisant donc à un champ de 7 bits.

U.3.1.6.2 Règles relatives à la classification et au transfert d'informations

D'un point de vue de la mise en cache et du placement en tampon, une information peut être classée comme étant "constant", "static", "dynamic" ou "non-cacheable". Ces classifications sont décrites en 12.6.3 pour les attributs d'objets natifs. La même ligne directrice s'applique à la sélection de la mise en tampon pour les interfaces "publish" et tunnel pour les charges utiles étrangères.

Il convient que des informations "constant" ne soient pas transférées plus d'une fois entre des TN, excepté lorsque les copies locales sont perdues en raison de cycles de puissance, d'une réinitialisation, d'un effacement de cache ou d'une élimination de références aux informations.

Il convient que les informations "static" ne soient transférées plus d'une fois entre des TN, excepté dans les conditions indiquées pour les informations "constant" et lorsque les informations statiques ont été modifiées.

Il convient de transférer une information "dynamic" seulement entre des TN lorsque sa valeur a changé, sauf s'il est exigé plus souvent d'indiquer que la source ou la destination est encore active.

Les informations "non-cacheable" peuvent être transférées entre les TN sur chaque demande.

U.3.1.6.3 Interface publish/subscribe

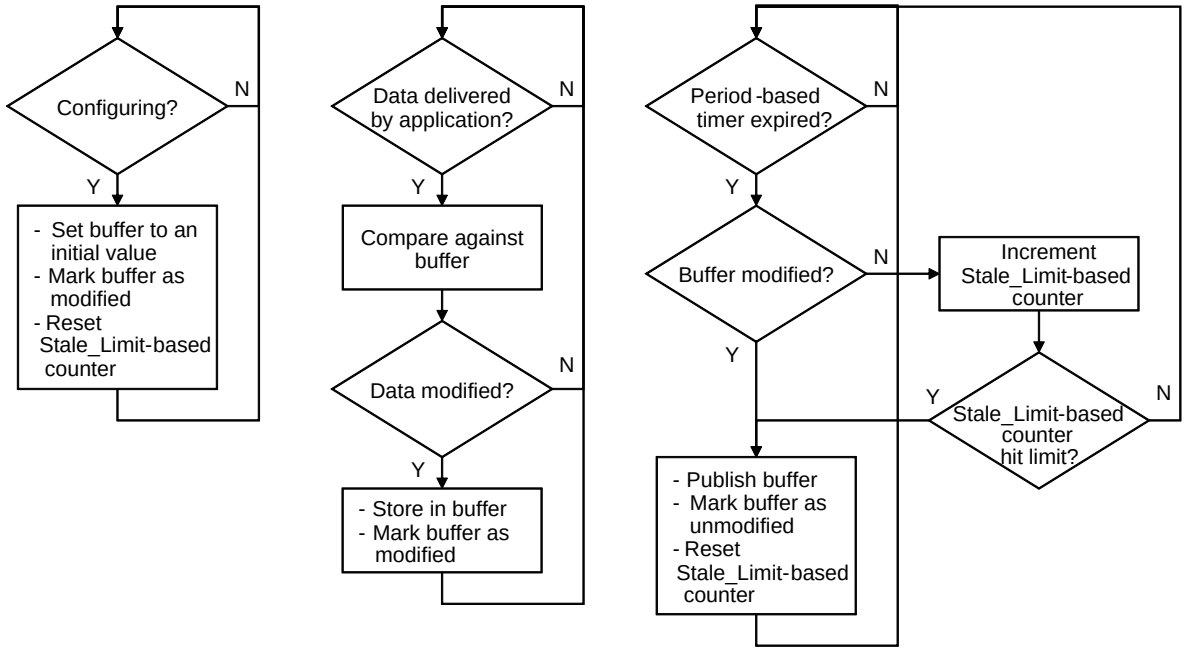
L'objet tunnel peut être mis en œuvre pour fournir des moyens d'accomplir la messagerie P/S pour la mise à jour des informations dynamiques.

Les organigrammes de la Figure U.20, de la Figure U.21 et de la Figure U.22 décrivent le comportement des objets TUN éditeurs et abonnés qui utilisent le tampon. Le comportement

décrit l'accord de transfert de messages de base entre un éditeur et un abonné en fonction des configurations des attributs des objets TN.

NOTE L'interprétation et les actions pour des données initiales, périmées et répétées sont basées sur la mise en œuvre, comme l'est l'algorithme CoSt.

La connexion d'éditeur P/S fonctionne conformément à la Figure U.20 lorsque les mises à jour de CoSt sont configurées.

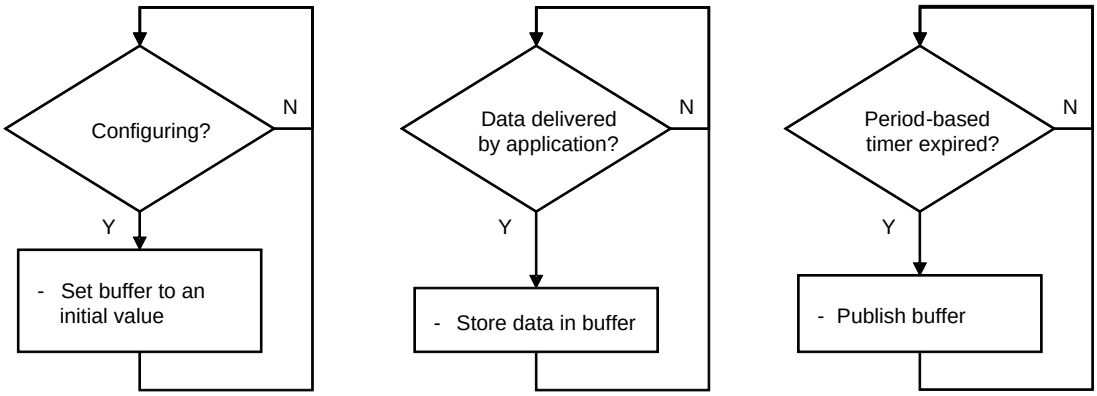


Légende

Anglais	Français
Configuring?	En cours de configuration?
Y	O
Set buffer to an initial value Mark buffer as modified Reset Stale_Limit-based counter	Mettre le tampon à une valeur initiale Marquer le tampon comme étant modifié Réinitialiser le compteur basé sur Stale_Limit
Data delivered by application?	Données livrées par l'application?
Compare against buffer	Comparer au tampon
Data modified?	Données modifiées?
Store in buffer Mark buffer as modified	Stocker dans le tampon Marquer le tampon comme étant modifié
Period-based timer expired?	Le temporisateur basé sur la période a expiré?
Buffer modified?	Tampon modifié?
Publish buffer Mark buffer as unmodified Reset Stale_Limit-based counter	Editer le tampon Marquer le tampon comme étant inchangé Réinitialiser le compteur basé sur Stale_Limit
Increment Stale_Limit-based counter	Incrémenter le compteur basé sur Stale_Limit
State_Limit-based counter hit limit?	Compteur basé sur Stale_Limit a atteint la limite?

Figure U.20 – Organigramme de CoSt d'éditeur de P/S

La connexion d'éditeur P/S fonctionne conformément à la Figure U.21 lorsque des mises à jour périodiques sont configurées.

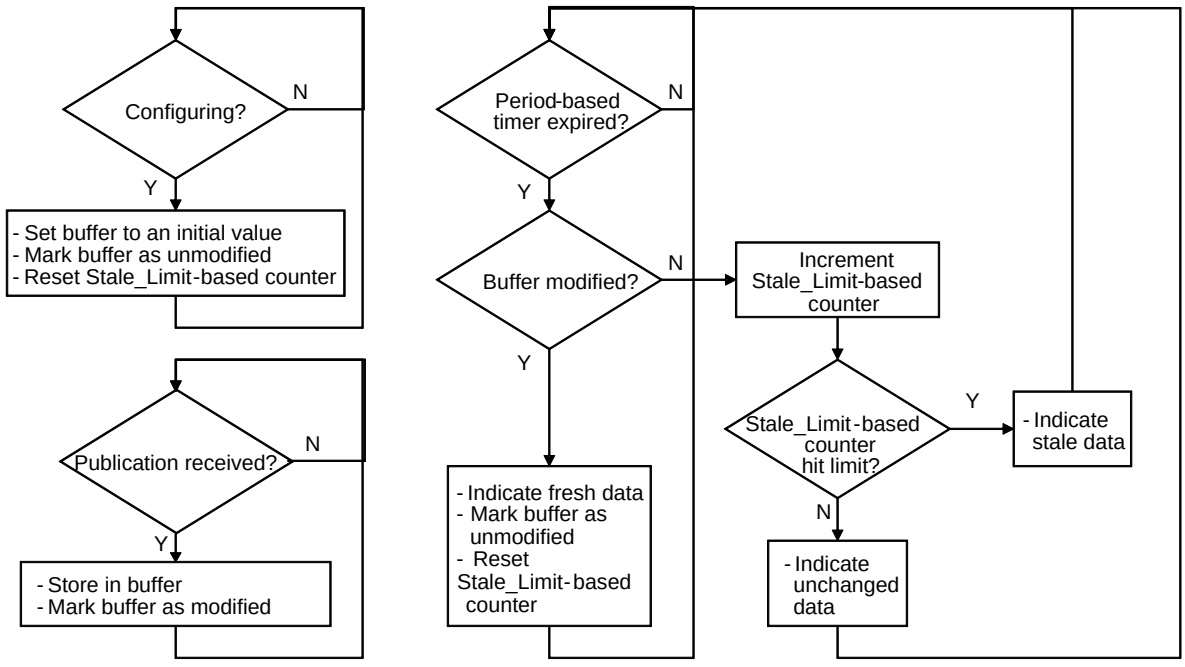


Légende

Anglais	Français
Configuring ?	En cours de configuration?
Y	O
Set buffer to an initial value	Mettre le tampon à une valeur initiale
Data delivered by application?	Données livrées par l'application?
Store data in buffer	Stocker les données dans le tampon
Period-based timer expired?	Le temporisateur basé sur la période a expiré?
Publish buffer	Editer le tampon

Figure U.21 – Organigramme de mises à jour périodiques d'éditeur de P/S

La connexion d'abonné P/S fonctionne conformément à la Figure U.22 lorsque des mises à jour périodiques ou de CoSt sont configurées.



Légende

Anglais	Français
Configuring ?	En cours de configuration?
Y	O

Anglais	Français
Set buffer to an initial value Mark buffer as unmodified Reset Stale_Limit-based counter	Mettre le tampon à une valeur initiale Marquer le tampon comme étant inchangé Réinitialiser le compteur basé sur Stale_Limit
Publication received?	Publication reçue?
Store in buffer Mark buffer as modified	Stocker dans le tampon Marquer le tampon comme étant modifié
Period-based timer expired?	Le temporisateur basé sur la période a expiré?
Buffer modified?	Tampon modifié?
Indicate fresh data Mark buffer as unmodified Reset Stale_Limit-based counter	Indiquer des données fraîches Marquer le tampon comme étant inchangé Réinitialiser le compteur basé sur Stale_Limit
Increment Stale_Limit-based counter	Incrémenter le compteur basé sur Stale_Limit
Stale_Limit-based counter hit limit?	Compteur basé sur Stale_Limit a atteint la limite?
Indicate unchanged data	Indiquer des données inchangées
Indicate stale data	Indiquer des données périmées

Figure U.22 – Organigramme de mises à jour périodiques et de CoSt communes d'abonné P/S

U.3.1.6.4 Interface de tunnel

L'objet tunnel peut être mis en œuvre pour fournir des moyens d'accomplir une messagerie d'interface tunnel tamponnée et non tamponnée. La messagerie non tamponnée d'interface tunnel fournit la prise en charge pour le transfert inconditionnel des informations "non-cacheable" et "constant". La messagerie tamponnée d'interface tunnel fournit la prise en charge pour la mise en tampon et le transfert contingent d'informations "static" et "dynamic".

U.3.1.7 Réponses multiples du serveur

Certaines demandes C/S reçoivent plusieurs réponses. Une raison est que la demande exige un traitement prolongé et une réponse immédiate est envoyée, qui indique que la demande a été reçue et que la réponse réelle sera envoyée après que le traitement aura été parachevé. Cela s'appelle, dans certains protocoles, une réponse différée. Dans d'autres cas, le serveur fournit les mises à jour complémentaires au fil du temps pour satisfaire à la demande initiale. Certains protocoles recueillent, de cette manière, des variables de processus ou des informations historiques.

Les interfaces tamponnées et non amorties de C/S prennent en charge plusieurs réponses d'application à ces fins. Dans le cas de la réponse tamponnée, le tampon de lecture maintient la réponse la plus récente. Le client reçoit une indication sur chaque réponse.

U.3.1.8 Mapping d'adresses d'objet tunnel

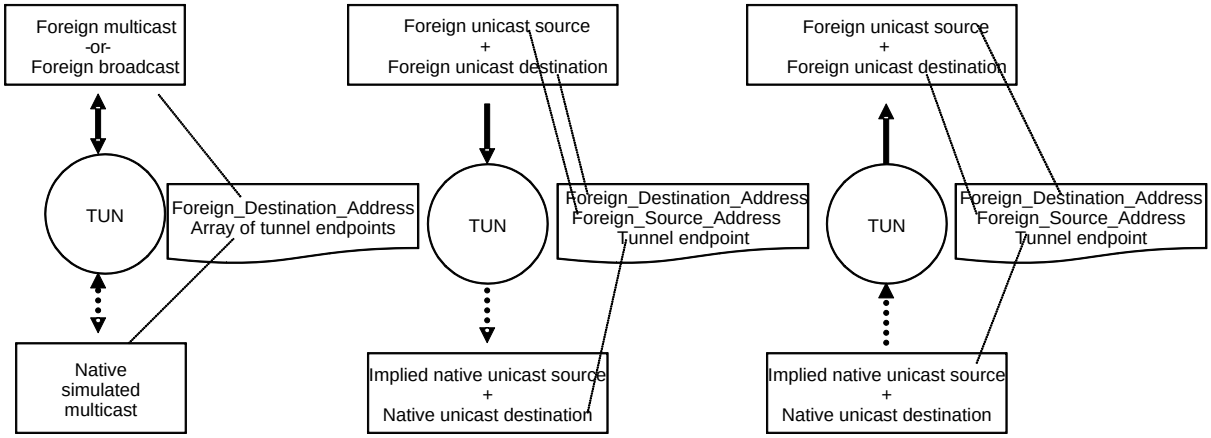
L'objet TUN peut être mis en œuvre pour contenir trois champs d'adresse (Foreign_Destination_Address, Foreign_Source_Address, et Array of Tunnel endpoints) qui sont utilisées dans la conversion entre adresses étrangères et adresses natives.

Telles que montrées à la Figure U.23, les adresses de multidiffusion étrangères et les adresses de diffusion requièrent une conversion en adresses et messagerie natives.

Le premier cas est celui dans lequel la multidiffusion ou la diffusion a son origine sur le réseau étranger. Sachant que plusieurs hôtes, protocoles ou applications peuvent partager un réseau sans fil tel que décrit dans le présent document, il n'est pas efficace d'envoyer une diffusion étrangère vers tous les appareils sans fil. Ainsi, la diffusion étrangère dans le réseau

sans fil utilise la multidiffusion simulée vers un groupe limité. L'objet TUN est utilisé pour simuler la livraison en multidiffusion (messagerie "un à plusieurs") en maintenant une liste d'adresses de monodiffusion (matrice de points d'extrémité de tunnel) et en utilisant une séquence de livraisons en monodiffusion.

Le deuxième cas est celui où la multidiffusion ou la diffusion a son origine sur le réseau sans fil et elle est destinée au réseau étranger. Une seule APDU est livrée du réseau sans fil et est traitée par un convertisseur de protocoles pour générer une PDU de multidiffusion ou de diffusion sur le réseau étranger.



Légende

Anglais	Français
Foreign multicast or foreign broadcast	Multidiffusion étrangère ou diffusion étrangère
Foreign unicast source + foreign unicast destination	Source de monodiffusion étrangère + destination de monodiffusion étrangère
Native simulated multicast	Multidiffusion simulée native
Implied native unicast source + native unicast destination	Source de monodiffusion native + destination de monodiffusion native implicites

Figure U.23 – Mappings d'adresses réseau

La Figure U.23 montre également une paire de conversions d'adresses de monodiffusion.

Le premier cas convertit une paire d'adresses source et destination étrangères en une paire d'adresses natives. Le second cas convertit une paire d'adresses source et destination natives en une paire d'adresses étrangères. Foreign_Destination_Address et Foreign_Source_Address sont utilisés tous les deux. Seules sont nécessaires les informations relatives aux adresses issues d'un seul point d'extrémité de tunnel, car l'objet TUN a accès à sa propre adresse native pour une utilisation dans les champs source ou destination. La définition de source et de destination étrangères dépend du sens du transfert.

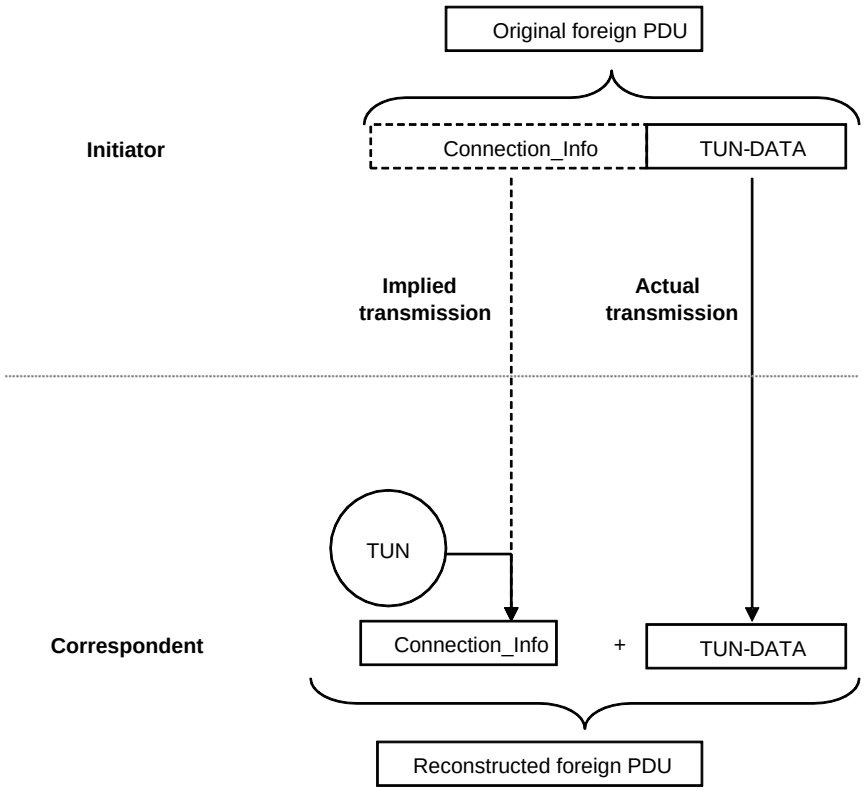
U.3.1.9 Informations relatives à une connexion et à une transaction

Les objets TN fonctionnent comme des points d'extrémité d'initiateur (demande d'éditeur et de tunnel) et des points d'extrémité correspondants (réponse d'abonné et de tunnel). La conversion de protocoles envoie le contenu étranger sous la forme de TUN-DATA entre les points d'extrémité. Sachant que la plupart des protocoles hérités ne sont pas optimisés pour la communication sans fil à faible consommation d'énergie, divers mécanismes sont disponibles pour augmenter l'efficacité.

Lorsqu'un convertisseur de protocoles tunnellise une PDU étrangère, il n'est pas efficace d'envoyer à répétitions les parties statiques de la PDU étrangère entre les points d'extrémité. De telles informations statiques comprennent des préambules et l'adressage fixe secondaire, tel que les identificateurs d'unités logiques. Tels que montrés à la Figure U.24, les objets

TUN fournissent un mécanisme générique (Connection_Info) pour la mise à disposition d'informations statiques à la réception de PDU étrangères sans transfert sans fil par message des informations statiques.

NOTE La PDU étrangère variera en fonction de la conversion de protocole étranger spécifique. La plupart des protocoles de bus de terrain formeront des DPDU pour une livraison directe sur une liaison locale. Par contraste, les protocoles basés sur IP forment habituellement des NPDU, où une encapsulation finale est réalisée par un protocole de résolution d'adresse.



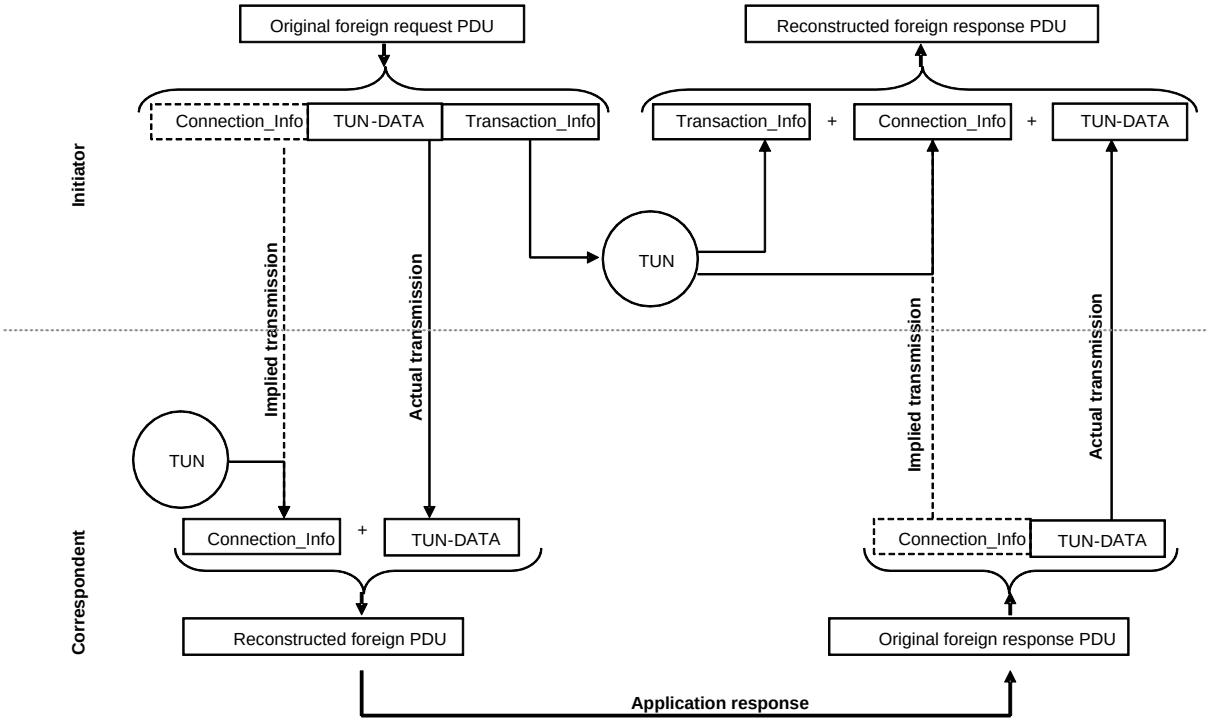
Légende

Anglais	Français
Original foreign PDU	PDU étrangère d'origine
Initiator	Initiateur
Implied transmission	Emission impliquée
Actual transmission	Emission réelle
Correspondent	Correspondant
Reconstructed foreign PDU	PDU étrangère reconstruite

Figure U.24 – Utilisation de Connection_Info dans la conversion de protocoles

Lorsqu'un convertisseur de protocole accomplit une transaction, il n'est pas efficace de transporter des informations spécifiques à une transaction qui sont seulement utilisées pour identifier la transaction au niveau de l'initiateur. De telles informations comprennent des informations pour relier la demande d'origine à la réponse, lorsque la connaissance du point d'extrémité peut être utilisée. Tels que montrés à la Figure U.25, les objets TUN fournissent un mécanisme générique (Transaction_Info) pour mettre à disposition les informations spécifiques à une transaction sans transporter le surdébit dans le transfert sans fil.

Connection_Info et Transaction_Info peuvent être utilisés simultanément tous les deux.



Légende

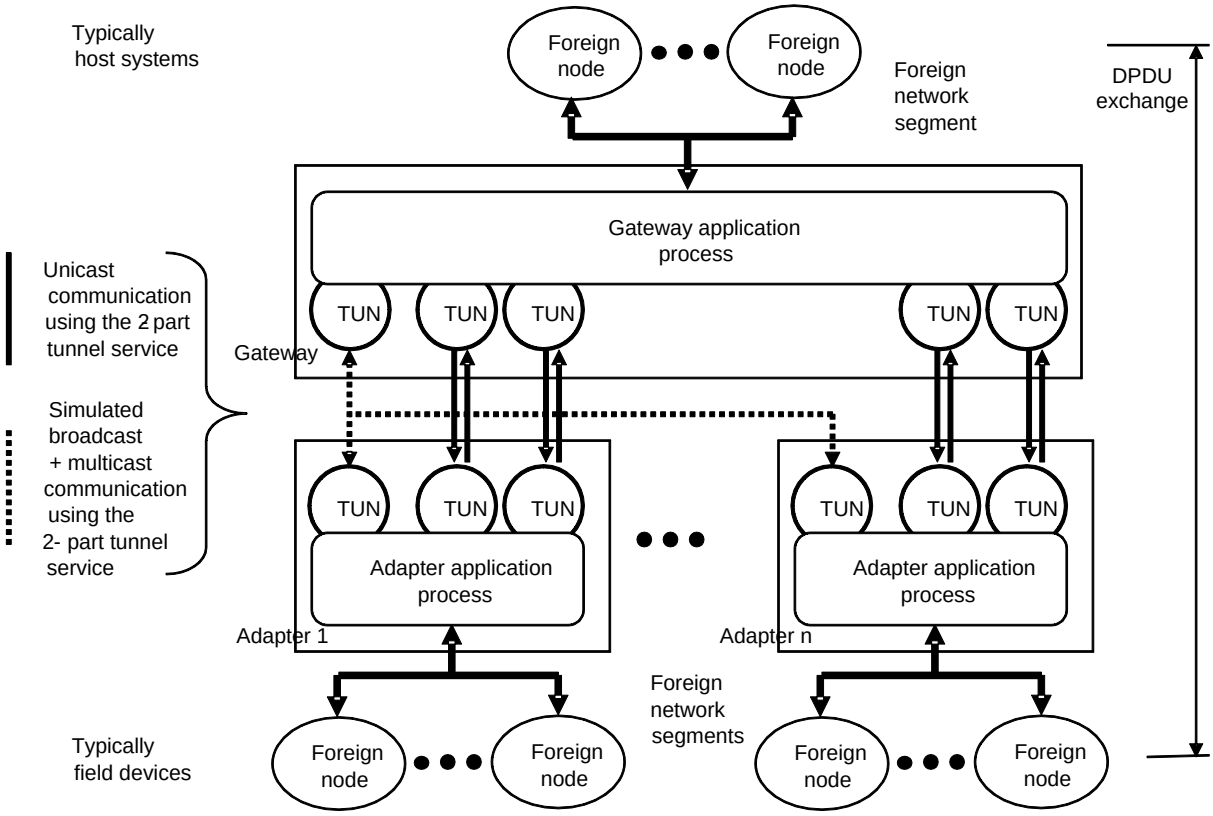
Anglais	Français
Original foreign request PDU	PDU de demande étrangère d'origine
Reconstructed foreign response PDU	PDU de réponse étrangère reconstruite
Initiator	Initiateur
Implied transmission	Emission implicite
Actual transmission	Emission réelle
Correspondent	Correspondant
Reconstructed foreign PDU	PDU étrangère reconstruite
Original foreign response PDU	PDU de réponse étrangère d'origine
Application response	Réponse d'application

Figure U.25 – Utilisation de Transaction_Info dans la conversion de protocoles

U.3.1.10 Mécanisme de tunnellisation interopérable

U.3.1.10.1 Vue d'ensemble

L'Annexe U décrit un mécanisme de communication pour que les nœuds de réseau étrangers communiquent à travers un réseau sans fil par l'intermédiaire de passerelles et d'adaptateurs. Ce mécanisme permet le développement, indépendant vis-à-vis de tout fournisseur, des passerelles et des adaptateurs en mettant en œuvre un sous-ensemble restreint des caractéristiques de communication au sein de la présente norme. La communication interopérable est réalisée par l'utilisation d'un mécanisme de tunnellisation contrainte. Les passerelles et les adaptateurs servent à interconnecter deux ou plus de deux segments de réseau étrangers en pontant les DPDU de protocole étranger par l'intermédiaire du réseau sans fil tel que décrit à la Figure U.26. Les processus d'application de passerelle et d'adaptateur utilisent les objets et les interfaces tunnel de l'AL pour l'échange de DPDU de monodiffusion étrangères et des DPDU de diffusion/multidiffusion étrangères. Le mécanisme par lequel les processus d'application de passerelle et d'adaptateur échangent ces DPDU avec des nœuds étrangers n'est pas spécifié par la présente norme.



Légende

Anglais	Français
Typically host systems	Systèmes typiquement hôtes
Foreign node	Nœud étranger
Foreign network segment	Segment de réseau étranger
DPDU exchange	Echange de DPDU
Gateway application process	Processus d'application de passerelle
Unicast communication using the 2-part tunnel service	Communication en monodiffusion utilisant le service tunnel à deux parties
Gateway	Passerelle
Simulated broadcast + multicast communication using the 2-part tunnel service	Communication simulée en diffusion et multidiffusion utilisant le service tunnel à deux parties
Adapter 1	Adaptateur 1
Adapter application process	Processus d'application d'adaptateur
Adapter n	Adaptateur n
Typically field devices	Appareils de terrain types

Figure U.26 – Vue d'ensemble d'un mécanisme de tunnellisation interopérable

U.3.1.10.2 Placement d'objet tunnel

Un ou plusieurs nœuds de réseau étrangers, individuellement adressables par une adresse de DPDU en monodiffusion, peuvent exister derrière une passerelle ou un adaptateur. Un nœud de réseau étranger derrière une passerelle ou un adaptateur peut exiger la communication avec un nœud de réseau étranger associé placé derrière une autre passerelle ou un autre adaptateur.

Pour chaque passerelle et chaque adaptateur associés, un objet tunnel est disposé et configuré pour transporter des DPDU étrangères adressées en diffusion et multidiffusion, une

pour un premier segment de réseau étranger associé et une pour chaque segment de réseau étranger associé supplémentaire.

Pour chaque paire de nœuds de réseau étrangers associés, une paire d'objets tunnel est disposée et configurée pour transporter des DPDU adressées en monodiffusion, une dans la passerelle ou dans l'adaptateur pour un premier nœud associé et une dans la passerelle ou dans l'adaptateur pour un second nœud associé.

U.3.1.10.3 Configuration d'objet tunnel

Le fonctionnement de tunnel est commandé comme décrit à l'Article 12. Les objets tunnel sont configurés par l'intermédiaire des valeurs de réglage des attributs. Les modifications apportées à la configuration doivent être correctement ordonnancées en positionnant l'attribut Protocol et en surveillant l'attribut Status.

Les paires d'objets tunnel en monodiffusion sont configurées comme suit:

- L'attribut Flow_Type est configuré pour un tunnel à deux parties.
- Les attributs Array of Tunnel endpoints sont configurés pour un seul élément d'adresse, où chaque objet tunnel dans la paire adresse l'autre objet tunnel dans la paire en question.
- Les attributs Connection_Info[ ] et Transaction_Info[ ] ne sont pas utilisés.
- Les attributs Update_Policy, Phase, Period et Stale_Limit ne sont pas utilisés.

Pour les objets tunnel de monodiffusion, l'attribut Foreign_Destination_Address de chaque objet tunnel local est mis à l'adresse de DPDU de l'appareil étranger associé derrière la passerelle distante ou l'adaptateur et l'attribut Foreign_Source_Address de chaque objet tunnel local sont mis à l'adresse de DPDU de l'appareil étranger associé derrière la passerelle locale ou l'adaptateur.

Les objets tunnel dans l'ensemble d'objets tunnel de diffusion/multidiffusion sont configurés comme suit:

- L'attribut Flow_Type est configuré pour un tunnel à deux parties.
- Les attributs Array of Tunnel endpoints sont configurés pour un ou plusieurs éléments d'adresse, où chaque objet tunnel dans l'ensemble adresse tous les autres objets tunnel dans l'ensemble en question.
- Les attributs Connection_Info[ ] et Transaction_Info[ ] ne sont pas utilisés.
- Les attributs Update_Policy, Phase, Period et Stale_Limit ne sont pas utilisés.

Pour des objets tunnel de diffusion/multidiffusion, l'attribut Foreign_Destination_Address et l'attribut Foreign_Source_Address sont mis à une valeur égale.

L'utilisation de l'attribut Foreign_Source_Address et de l'attribut de Foreign_Destination_Address permet que des passerelles et des adaptateurs utilisant le mécanisme de tunnellisation interopérable soient strictement configurés par la configuration des objets tunnel.

Les passerelles et les adaptateurs associés peuvent envoyer et recevoir des DPDU étrangères issues soit de versions identiques, soit de versions interopérables du même protocole étranger. Pour permettre l'état de marche après que les autres attributs ont été configurés, l'attribut Protocol est configuré le dernier et est configuré à la même valeur dans tous les objets tunnel associés à tous les segments de réseau étrangers relatifs connexes sur le sous-réseau D. La valeur finale de l'attribut Protocol est mise conformément à l'Annexe K. Les processus d'application de passerelle et d'adaptateur peuvent rapporter le Status d'objet tunnel = 2 (configuration échouée) si une tentative est faite de configurer un tunnel avec un protocole non pris en charge.

Un protocole étranger compatible peut être capable de prendre en charge la temporisation imposée par les mécanismes sans fil, soit de façon inhérente, soit par configuration. L'échange de DPDU étrangère peut ne pas être la méthode de tunnel la plus efficace, mais il s'assure que des informations suffisantes sont disponibles pour traiter le paquet au sein des processus d'application de passerelle et d'adaptateur. Il assure également que plusieurs fournisseurs acheminent les mêmes informations entre les processus d'application de passerelle et d'adaptateur. Il assure également que des information suffisantes sont disponibles dans le processus d'application de passerelle et d'adaptateur pour relier les demandes et les réponses C/S. L'adressage est également transporté et permet à plusieurs appareils de réseau étrangers de s'asseoir derrière chaque passerelle ou chaque adaptateur.

U.3.1.10.4 Fonctionnement de tunnel

Les DPDU de réseau étrangères peuvent être livrées vers les processus d'application de passerelle et d'adaptateur de l'une de deux manières, soit par l'intermédiaire d'un objet tunnel, soit à partir d'une source étrangère à l'extérieur du réseau sans fil. La source extérieure sera habituellement un réseau câblé (et sa pile de protocoles associée) rattaché directement ou indirectement à la passerelle ou à l'adaptateur. En variante, les PDU peuvent être générées par le logiciel ou le firmware interagissant avec (ou intégré dans) le processus d'application de passerelle ou d'adaptateur directement. Dans un cas comme dans l'autre, l'échange de PDU tunnellisée entre les passerelles et les adaptateurs peut rester identique.

Les processus d'application de passerelle et d'adaptateur peuvent examiner l'adresse de destination de DPDU de réseau étrangère avant de transmettre la PDU sur le réseau sans fil. Les DPDU sans destination connue qui sont accessibles par l'intermédiaire des tunnels ne sont pas transmises.

Les processus d'application de passerelles et d'adaptateurs peuvent transmettre des DPDU de monodiffusion de protocoles étrangers vers les destinations d'adresses de DPDU qui sont accessibles par l'intermédiaire d'une paire reliée d'objets tunnel de monodiffusion.

Les processus d'application de passerelle et d'adaptateur transmettent les PDU de diffusion et de multidiffusion de protocoles étrangers par le tunnel de diffusion/multidiffusion qui existe au sein de chaque passerelle et chaque adaptateur associés, distribuant la même PDU vers une ou plusieurs destinations. La PDU n'est pas reprise en écho vers la source.

Les processus d'application de passerelle et d'adaptateurs peuvent utiliser des PDU d'établissement de groupe de multidiffusion issues de l'intérieur du protocole étranger, lorsque de telles PDU existent, afin de limiter la portée de la distribution.

Sachant que la génération de plusieurs copies du même message est presque certaine de se produire, le protocole étranger peut tolérer le biais de temporisation.

U.3.1.10.5 Fonctionnement efficace

Il convient que les protocoles étrangers qui utilisent le mécanisme de tunnellisation interopérable ramènent les échanges des PDU au minimum qui est acceptable au protocole étranger et à ses applications. Cela est accompli en allongeant des périodes et des temporisations de mise à jour pour la mise à jour périodique. Cela est également accompli par élimination de transfert redondant d'informations statiques en maintenant des copies locales.

U.3.2 Transfert en masse

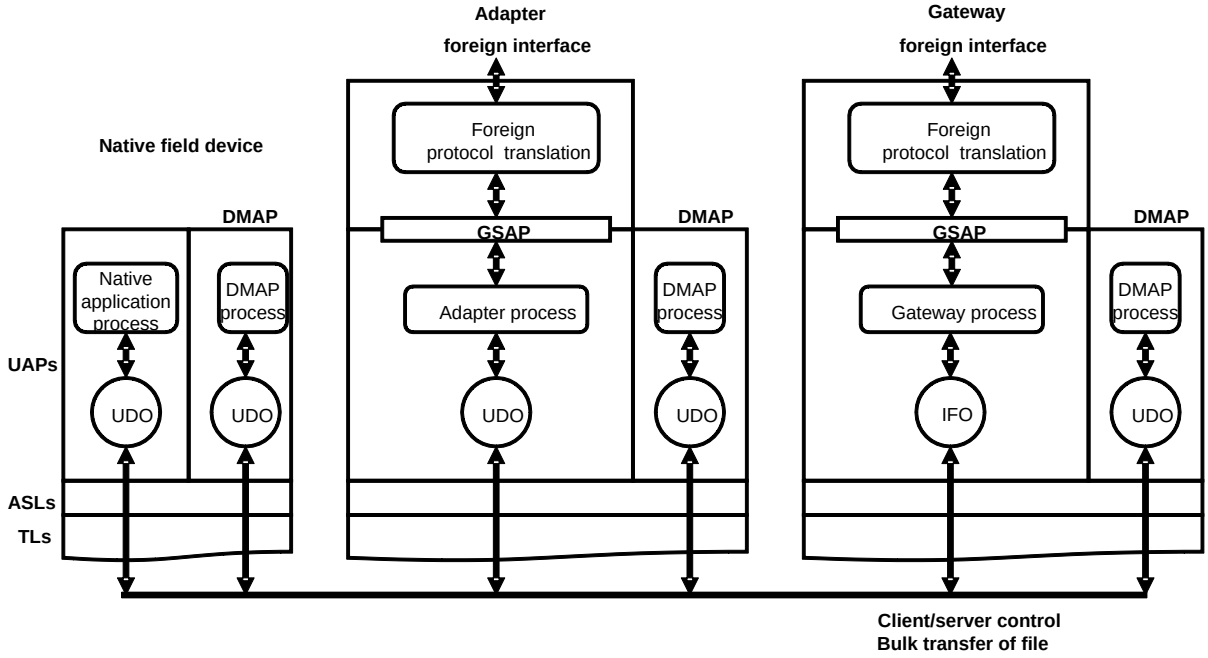
Le transfert d'éléments volumineux est accompli par l'intermédiaire d'objets téléchargement montant/téléchargement descendant (les UDO), tels que montrés à la Figure U.27. Les transferts d'éléments volumineux sont utiles pour des mises à jour de firmware, le transfert de tampons d'échantillons volumineux tels que les formes d'onde capturées, et pour la configuration générale. Un UDO représente un seul élément qui peut être transféré dans l'un ou l'autre sens (téléchargé en montant ou téléchargé en descendant) vers/depuis une autre

application. L'élément devant être transféré existe à l'emplacement de l'UDO. Les objets interface (IFO) agissent comme des clients pour initier des transferts. Le protocole de transfert fournit le placement en tampon, le contrôle de flux ainsi qu'une livraison garantie et ordonnée. Les convertisseurs de protocole ont accès à l'UDO par l'intermédiaire du GIAP.

Des éléments sont associés à une chaîne qui peut être utilisée pour coder des informations relatives à l'identification et à la révision spécifiques à un élément. Des systèmes de gestion d'actifs peuvent être construits pour surveiller des révisions pour des industries régulées et pour effectuer des sauvegardes de secours et restaurer des éléments par des moyens génériques, sans connaissance du contenu de l'élément. Les convertisseurs de protocole peuvent également transférer des éléments volumineux en passant par des protocoles étrangers par l'intermédiaire de la tunnellisation, mais cela empêche la gestion d'actifs indépendant vis-à-vis de tout protocole.

Des applications finales sont censées comprendre le contenu de l'élément transféré et la manière de l'appliquer. Des dispositions existent (en fonction des capacités d'appareil) pour demander l'utilisation de l'élément (en altérant possiblement le comportement en temps d'exécution) et pour le stockage de l'élément dans une mémoire non volatile.

Les UDO et le protocole de transfert en masse de téléchargement montant et de téléchargement descendant sont décrits en 12.15.2.4.



Légende

Anglais	Français
Adapter foreign interface	Interface étrangère d'adaptateur
Gateway foreign interface	Interface étrangère de passerelle
Native field device	Appareil d'interface natif
Foreign protocol translation	Conversion de protocole étranger
Native application process	Processus d'application natif
DMAP process	Processus de DMAP
Adapter process	Processus d'adaptateur
Gateway process	Processus de passerelle
Client/server control	Commande client/serveur
Bulk transfer of file	Transfert en masse de fichier

Figure U.27 – Modèle de transfert en masse

U.3.3 Alertes

Les alertes peuvent être générées par plusieurs des objets définis dans la présente norme. Certains objets résident au sein d'UAP d'appareil, alors que d'autres résident dans le DMAP comme objets de gestion pour chaque couche.

Les alertes au sein d'un appareil sont consolidées au sein de l'objet de gestion de rapports d'alertes (ARMO). Chaque appareil a un seul ARMO qui réside au sein du DMAP. Toutes les alertes au sein d'un appareil sont commodément consolidées dans ce seul emplacement.

L'ARMO dans chaque DMAP est responsable de rapporter des alertes par l'intermédiaire d'une interface AlertReport à un objet récepteur d'alertes (ARO). L'ARO acquitte la réception d'alerte par l'intermédiaire de l'interface AlertAcknowledge. Ce transfert se produit indépendamment du traitement réel des alertes.

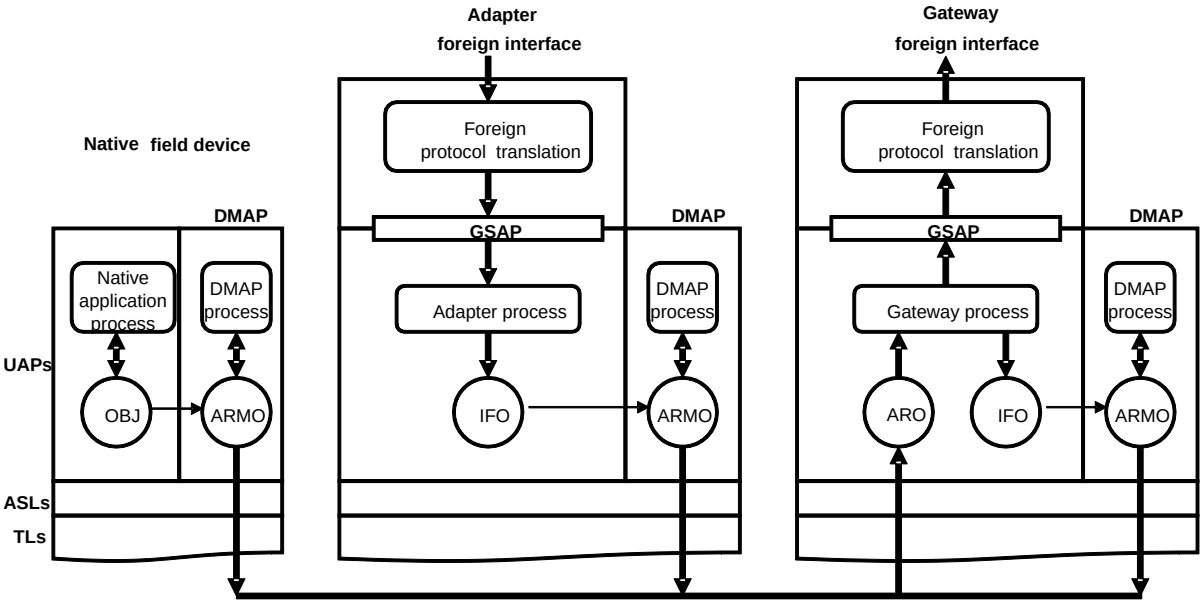
Les alertes s'inscrivent dans quatre catégories:

- processus;
- appareil;
- réseau; et
- sécurité.

Chaque catégorie peut être livrée par un ARMO à un ARO différent. Ainsi, un seul ARO simple pourrait recueillir toutes les alertes de processus à travers un réseau entier, ou un ensemble d'ARO peut être utilisé, avec chaque ARO recueillant seulement une seule catégorie d'alertes. Si chaque ARO recueille un seul type d'alerte, la collecte de toutes les alertes exige quatre ARO.

La passerelle contient un ou plusieurs ARO qui permettent la collecte, la production de rapports et la gestion des alertes.

Les convertisseurs de protocole ont accès à des alertes par l'intermédiaire du GIAP et peuvent les gérer, conformément au modèle d'alerte montré à Figure U.28. Les interfaces d'abonnement permettent la sélection des alertes par l'intermédiaire du GIAP.



Légende

Anglais	Français
Adapter foreign interface	Interface étrangère d'adaptateur

Anglais	Français
Gateway foreign interface	Interface étrangère de passerelle
Native field device	Appareil d'interface natif
Foreign protocol translation	Conversion de protocole étranger
Native application process	Processus d'application natif
DMAP process	Processus de DMAP
Adapter process	Processus d'adaptateur
Gateway process	Processus de passerelle

Figure U.28 – Modèle d'alerte

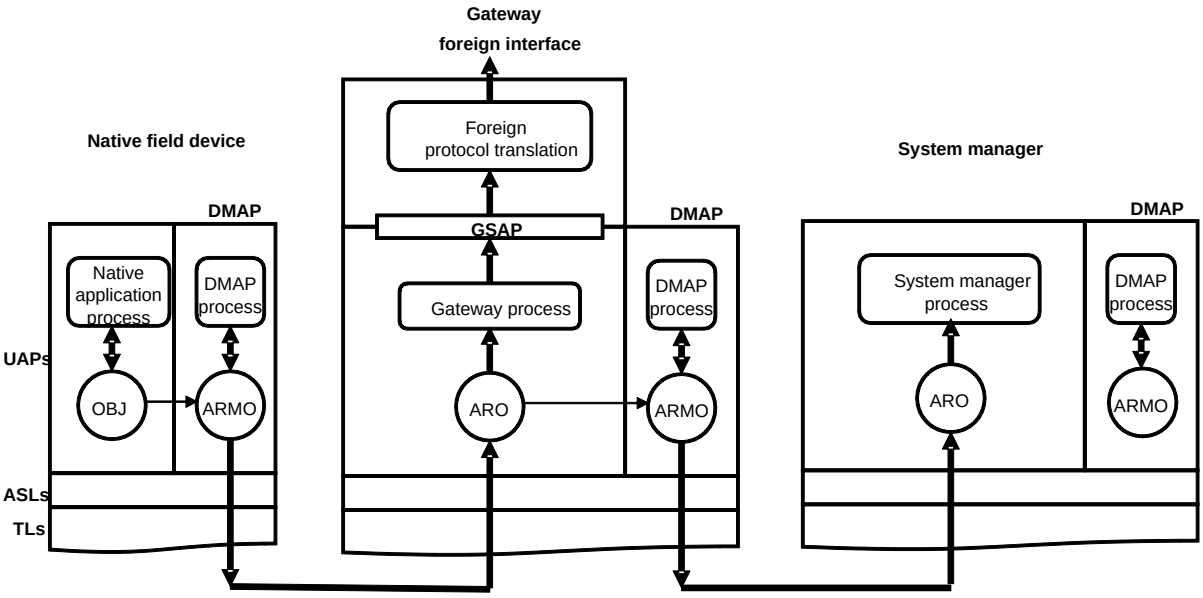
Les alertes s'inscrivent dans deux classes, à savoir alarmes et événements. Les événements sont informationnels et génèrent des messages d'événement par l'intermédiaire du GIAP. Les alarmes ont des états et requièrent des actions spécifiques à une alarme pour éliminer les alarmes. Habituellement, la messagerie client/serveur est utilisée pour entreprendre ces actions.

Les applications de passerelle et d'adaptateur peuvent également générer des alertes natives à partir des instances d'IFO. Cela permet à des convertisseurs de protocoles de générer des alertes au sein du contexte d'un système de gestion d'alertes normalisées.

Dans certaines circonstances, l'état d'alertes peut être perdu au niveau de l'ARO, comme lorsqu'une passerelle est réinitialisée ou remplacée. Dans un tel cas, les ARMO d'origine ne contiendront plus d'informations relatives à des événements, mais maintiendront les informations d'état relatives aux alarmes. Une procédure de récupération d'alarme peut être initiée afin de récupérer l'état d'alarme du système.

La présente norme ne prend pas en charge les alertes en multidiffusion. Il en résulte que les mêmes alertes ne peuvent pas être acheminées à la passerelle et au gestionnaire de système si elles ne sont pas contiguës physiquement. Des alertes réseau et sécurité sont actuellement envoyées dans le gestionnaire de système par défaut. Des alertes processus et appareil peuvent être envoyées vers le rôle de passerelle.

Le modèle d'alerte ne pas prend en charge les alertes de multidiffusion. Les alertes réseau et les alertes sécurité sont potentiellement utiles dans une passerelle pour la transformation en messages d'erreur de protocoles étrangers génériques. Le gestionnaire de système est la destination par défaut pour ces alertes. Dans des configurations de système où le gestionnaire de système est connecté au WISN par l'intermédiaire de la passerelle, l'ARO dans la passerelle, lorsqu'il est configuré pour recueillir des alertes pour des besoins du réseau et de la sécurité, est capable de présenter à nouveau les alertes par l'intermédiaire de l'ARMO local au gestionnaire de système. C'est ce que montre la Figure U.29.



Légende

Anglais	Français
Gateway foreign interface	Interface étrangère de passerelle
Native field device	Appareil d'interface natif
Foreign protocol translation	Conversion de protocole étranger
System manager	Gestionnaire de système
Native application process	Processus d'application natif
DMAP process	Processus de DMAP
Gateway process	Processus de passerelle
System manager process	Processus de gestionnaire de système

Figure U.29 – Cascades d'alertes

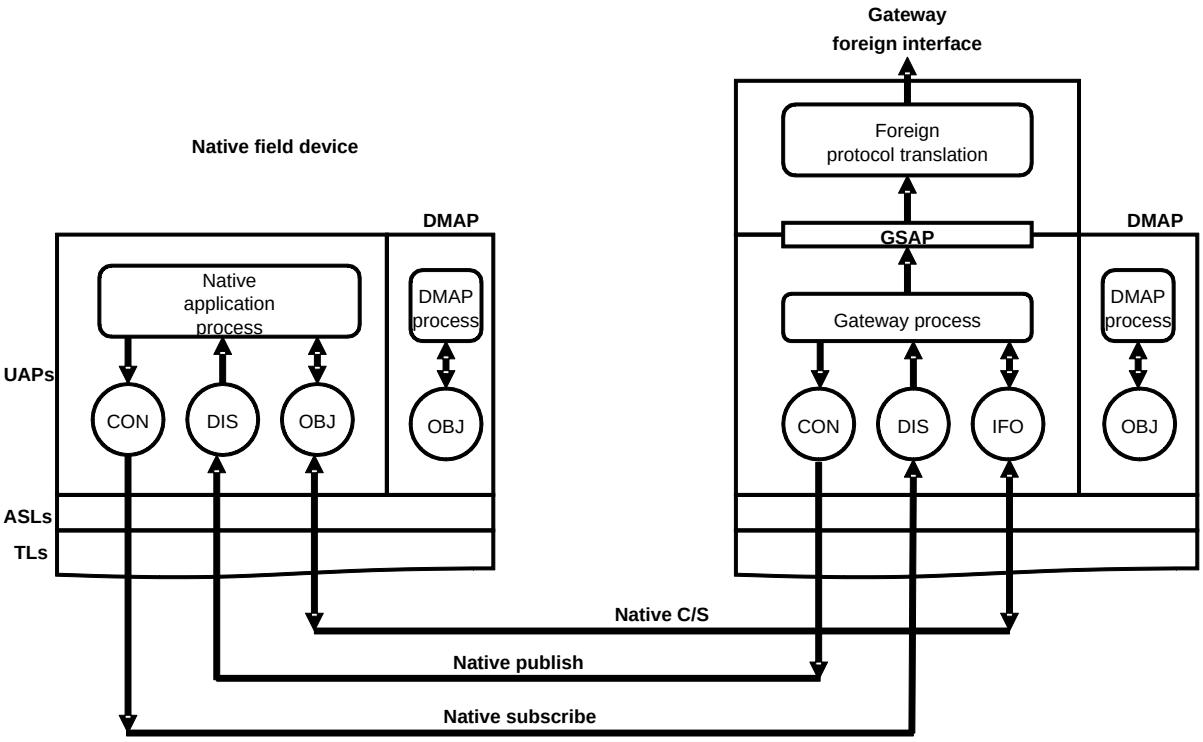
U.3.4 Accès P/S et client/serveur natif

La présente norme fournit des interfaces P/S et C/S par l'intermédiaire de l'ASL qui est utilisée pour interagir avec les objets natifs spécifiques à une application.

Pour les interfaces P/S, les objets concaténation (CON) et dispersion (DIS) sont utilisés comme des points d'extrémité.

Pour les interfaces C/S, deux points d'extrémité d'objet sont exigés afin d'utiliser ces interfaces. L'IFO peut agir comme un point d'extrémité pour ces interfaces au sein de passerelles. Toute autre application ou tout autre objet de gestion au sein du système peut agir comme étant l'autre point d'extrémité.

Comme montré à la Figure U.30, l'utilisation de ces objets permet à des convertisseurs de protocoles d'intégrer des appareils simples qui ne comportent pas de protocoles hérités.



Légende

Anglais	Français
Gateway foreign interface	Interface étrangère de passerelle
Native field device	Appareil d'interface natif
Foreign protocol translation	Conversion de protocole étranger
Native application process	Gestionnaire de système
DMAP process	Processus d'application natif
Gateway process	Processus d'adaptateur
Native C/S	C/S natif
Native publish	Editeur natif
Native subscribe	Abonné natif

Figure U.30 – Accès P/S et client/serveur natif

Dans une passerelle, les objets CON et DIS peuvent fournir le comportement de message tamponné pour l'opération de changement d'état.

Au sein d'une passerelle, l'IFO peut fournir le comportement de message tamponné tel que décrit pour la messagerie tunnel à quatre parties entre les objets tunnel pour les interfaces de lecture C/S. L'IFO peut utiliser la classification des attributs pour déterminer le comportement du placement en tampon. Les attributs "non-cacheable" ne sont pas tamponnés. Les attributs "constant" sont tamponnés. Le placement en tampon des attributs "static" et "dynamic" est déterminé par les exigences relatives à l'application.

Le convertisseur de protocoles peut utiliser l'adressage natif (Network_Address, Transport_Port, OID, et identificateur d'attribut) pour identifier les messages natifs.

NOTE La tunnellisation suppose que des messages de protocoles étrangers sont transférés entre des points d'extrémité. À ce titre, les adresses étrangères sont associées aux messages et utilisées pour la démolition et la reconstruction des messages afin d'éviter le transfert. Aucune hypothèse de cette nature n'est posée pour la messagerie native, où un flux de message biunivoque est moins susceptible d'exister.

U.3.5 Gestion du temps

Le temps d'hôte peut être propagé par une passerelle vers un système sans fil, donnant au système hôte et aux appareils de terrain le même sens du temps (dans les limites des tolérances). Cela permet d'utiliser le temps d'hôte pour des besoins tels que le marquage temporel uniforme des alertes et la séquence de détermination d'événements qui recouvrent les appareils sans fil et les appareils câblés connectés à l'hôte. Sans synchronisation périodique au temps d'hôte, le système sans fil dérivera et, donc, la capacité de réajustement périodique est souhaitable. Le système d'hôte et le système sans fil peuvent être synchronisés à une source extérieure commune telle qu'une source de temps dérivée du GPS.

Pour propager le temps d'hôte, une passerelle peut accomplir une synchronisation périodique du temps dans un temps de sous-réseau D rattaché à une source externe en demandant des changements de temps par l'intermédiaire d'un DLMO.

Les convertisseurs de protocoles au sein d'une passerelle peuvent accéder à des fonctions de gestion du temps par l'intermédiaire des services de GIAP. Les convertisseurs de protocoles sont responsables d'accéder à des sources de temps extérieures et de convertir des protocoles et des formats de temps. Le temps de réseau est représenté dans le format TAI, comme décrit en 5.6.

Une DL configurée comme une base d'horloge est utilisée pour propager les informations de synchronisation du temps vers un sous-réseau D rattaché, comme décrit en 6.3.10.3. Chaque nœud contient un DMO au sein de son DMAP. Le DMO contient les attributs DL_Subnet_Clock_Master_Role et DL_Subnet_Clock_Repeater_Role qui commandent la capacité d'un nœud à être une base d'horloge. L'attribution du rôle de base d'horloge est coordonnée avec le gestionnaire de système. L'appareil enregistre sa capacité à être une source de temps au cours du processus de rattachement du sous-réseau.

Le DLMO contient un attribut appelé TaiTime qui rapporte un temps courant et un autre attribut appelé TaiAdjust pour réajuster le temps. Le DLMO est utilisé pour réajuster le temps du sous-réseau D.

Une ou plusieurs DL peuvent être associées à une passerelle. Dans une mise en œuvre, les DL sont intégrées au sein de la passerelle. Dans une autre mise en œuvre, les DL se situent au sein de routeurs dorsaux et sont séparées de la passerelle, ajoutant des retards indéterminés. Chaque mise en œuvre peut considérer les implications du retard associé à l'accès d'objets d'DL pour accomplir la synchronisation.

U.3.6 Sécurité

Des ensembles d'appareils sans fil sont reliés à un hôte étranger par l'intermédiaire d'une passerelle. La passerelle et les appareils sans fil sont censés appartenir à un groupe commun de sécurité. La sécurité pour ce groupe peut être établie par la configuration de sécurité de couche MAC et/ou de couche TL, comme décrit à l'Article 7. L'établissement des valeurs de réglage de sécurité communes est un préalable à la communication entre les points d'extrémité de communication de conversion de protocoles.

Les protocoles étrangers de bus de terrain communs n'ont pas de capacités de sécurité. Cela n'exclut pas l'extension des protocoles sécurisés vers le domaine de la présente norme. Il est de la responsabilité des convertisseurs de protocoles étrangers, tant dans les passerelles que dans les adaptateurs, d'agir comme des applications de confiance dans l'extension de la sécurité de bout en bout des protocoles étrangers. Cela peut être réalisé par l'utilisation de sécurité native ou par l'intermédiaire d'échanges tunnellisés.

U.3.7 Configuration

Pour les passerelles qui mettent en œuvre des interfaces internes telles que les interfaces GSAP exemplaires, il convient que la configuration/capacité de l'entité de passerelle soit

rendue disponible en interne à des clients internes à la passerelle. La façon dont ces attributs opérationnels internes de passerelle sont rendus disponibles relève d'une initiative locale. Des attributs exemplaires que l'entité GSAP de la passerelle peut souhaiter présenter sont décrits dans le Tableau U.41. Par souci de commodité, l'attribut décrivant les conventions utilisées sont ceux qui sont utilisés par d'autres articles de la présente norme. Voir également l'exemple d'interface G_Read_Gateway_Configuration pour un exemple de la façon dont une interface peut être utilisée pour rendre ces informations disponibles à un client interne à une passerelle.

Tableau U.41 – Exemple d'attributs de gestion de configuration de passerelle

Nom du type d'objet normalisé: non applicable				
Identificateur du type d'objet normalisé: non applicable				
Nom de l'attribut	**Identificateur de l'attribut**	**Description de l'attribut**	**Informations sur les données de l'attribut**	**Description du comportement de l'attribut**
Max_Devices	11	Nombre maximal d'appareils pris en charge par la passerelle	Type: Unsigned16 Classification: Statique Accessibilité: Lecture seule (Read Only) Valeur par défaut: 0	Valeur dépendante de l'implémentation définie par la passerelle en fonction des ressources
Actual_Devices	12	Nombre actuel d'appareils connectés à la passerelle	Type: Unsigned16 Classification: Dynamique Accessibilité: Lecture seule (Read Only)	Hausses et baisses basées sur les appareils en communication avec la passerelle
Max_Leases	13	Nombre maximal de locations prises en charge par la passerelle	Type: Unsigned16 Classification: Statique Accessibilité: Lecture seule (Read Only)	Valeur dépendante de l'implémentation définie par la passerelle en fonction des ressources
Actual_Leases	14	Nombre actuel de locations connectés à la passerelle	Type: Unsigned16 Classification: Dynamique Accessibilité: Lecture seule (Read Only)	Hausses et baisses basées sur les locations. La complexité de l'appareil déterminera le nombre de locations exigé

U.3.8 Configuration et rattachement

Une passerelle est un appareil de réseau tel que décrit dans la présente norme et elle est configurée en utilisant les méthodes génériques décrites dans la présente norme.

Une passerelle qui communique vers des sous-réseaux D par l'intermédiaire de routeurs dorsaux peut fournir une méthode pour configurer la passerelle pour communiquer vers un sous-réseau D spécifique et vers des appareils spécifiques au sein du sous-réseau D en question par l'intermédiaire d'un routeur dorsal spécifique.

NOTE 1 Rien n'exclut des mises en œuvre plus dynamiques, telles qu'un algorithme de partage de charge qui assigne des appareils au meilleur BBR trouvé, ou des passerelles et des BBR qui se découvrent l'un l'autre, ou prennent en charge la redondance qui est fournie automatiquement lorsque les sous-réseaux D se chevauchent.

Une passerelle qui communique vers un ou plusieurs sous-réseaux D par des routeurs dorsaux inclut une méthode pour configurer la passerelle pour communiquer avec au moins un gestionnaire de système, lorsque le gestionnaire de système peut résider:

- dans la passerelle;
- sur une dorsale que la passerelle peut utiliser pour la communication; ou

- au sein d'un sous-réseau D que la passerelle peut utiliser pour la communication.

Une passerelle est un appareil de réseau (contenant une AL et une adresse IPv6) tel que décrit dans la présente norme et elle rejoint le réseau à la suite des méthodes de rattachement décrites dans la présente norme.

NOTE 2 Plusieurs variantes sont possibles, par exemple: une passerelle qui se rattache en envoyant une demande interne à un gestionnaire de système corésidant, ou en envoyant une demande de rattachement par l'intermédiaire d'une PhL locale, ou qui utilise indirectement une PhL de routeur dorsal ou qui envoie la demande de rattachement à travers la dorsale vers un gestionnaire de système.

Annexe V
(informative)

Conformité avec ETSI EN 300 328 v1.8.1

NOTE 1 Avec quelques modifications (comme l'ajout adéquat de quelques "doit"), on pourrait faire de cette Annexe V la base normative pour exploitation dans la CE. Dans ce cas, l'ETSI EN 300 328 devrait être promu de la Bibliographie aux références normatives de l'Article 2.

La norme ETSI EN 300 328 est un ensemble d'exigences qui interagissent de façon complexe, ce qui requiert des déclarations spécifique de comportement de fonctionnement. Il apparaît qu'il existe plusieurs modes d'exploitation différents sous lesquels un appareil conforme à la présente norme IEC peuvent être conformes à ces exigences, qui sont généralement plus favorables pour l'exploitation de la présente norme IEC que ceux d'autres ETSI EN comme l'ETSI EN 300 440.

Les appareils conformes avec l'ETSI EN 300 328 peuvent modifier leur mode d'exploitation de façon dynamique. Il est vraisemblable que la conformité de chaque mode d'exploitation potentiel devrait être évaluée de façon indépendante.

Selon la norme ETSI EN 300 328, IEEE 802.15.4 2,4 GHz DSSS est considéré comme une modulation large bande (WBM). Tel qu'utilisé dans la présente norme IEC, il est considéré également comme une modulation à étalement de spectre à saut de fréquence (FHSSM) lorsque le programme cyclique de saut de fréquence indique au moins 15 voies.

NOTE 2 Même en cas de WBM, une certaine quantité de sauts de fréquence est nécessaire pour éviter l'évanouissement de la bande étroite qui survient fréquemment en cas de durée supérieure à quelques ms. Ainsi, le saut de fréquence se produit qu'il soit revendiqué pour un fonctionnement en mode FHSSM relatif à la conformité avec la norme ETSI EN 300 328 ou non.

NOTE 3 La norme ETSI EN 300 328:2012 v1.8.1, 4.3.1.3.2 permet de bloquer les transmissions sur certaines des voies indiquées dans le programme de saut de fréquence (mise sur liste noire) mais ne permet pas de réduire le nombre de chaînes de sauts à moins de 15. Par conséquent, l'inclusion de moins de 15 voies dans une carte de voies qui détermine le cycle de saut de fréquence des voies nominalement actives signifie que les seuls régimes de fonctionnement possibles restants sont ceux sous WBM.

Dans la présente norme IEC, les initiateurs de transactions D qui activent la détection d'activité de voie CSMA/CA "listen before talk" (LBT) avant l'envoi de chaque DPDU Data satisfait aux exigences de la norme ETSI EN 300 328 pour l'adaptivité.

NOTE 4 Ces exigences sont ETSI EN 300 328:2012 v1.8.1, 4.3.1.6.1 (FHSSM) et 4.3.2.5.2.2.1 (WBM) et le texte lié. Bien que ces exigences mentionnent "modulation adaptative", la modulation référencée elle-même est fixée, pas adaptative; c'est l'utilisation de la modulation fixée qui est adaptative. Un terme plus adapté est donc "adaptivité".

Sous ETSI EN 300 328,

- Tx-sequence-time est le temps de démarrage de l'émetteur exigé pour envoyer une DPDU Data, qui est ≤ 4,256 ms. Dans certains cas, il s'agit également du temps de démarrage de l'émetteur pour l'envoi d'une DPDU ACK/NAK, qui est ≤ 1 ms;
- Tx-gap-time est l'intervalle minimal exigé de non-émission entre la fin d'une émission et le début de la suivante par le même appareil; et
- "temps de passage" (DT) est le temps nominal pendant lequel un initiateur de transaction D utilisant FHSSM conserve son émetteur ajusté à une voie avant de changer pour une autre voie.

NOTE 5 Tx-sequence-time et Tx-gap-time sont définis dans l'ETSI EN 300 328:2012 v1.8.1, 4.3.1.2 (FHSSM) et 4.3.2.3 (WBM). Le temps de passage qui s'applique à FHSSM est défini dans une certaine mesure dans l'ETSI EN 300 328:2012 v1.8.1, 3.1 sous "étalement de spectre à saut de fréquence" et dans l'ETSI EN 300 328:2012 v1.8.1, 4.3.1.3.1. Le temps de passage est obligatoirement au moins aussi grand que Tx-sequence-time.

L'ETSI EN 300 328:2012 v1.8.1, 4.3.2.2.2 (WBM) impose une limite de densité spectrale de la puissance pour une WBM de 10 dBm/MHz EIRP. A cause du spectre de la modulation IEEE 802.15.4 2,4 GHz DSSS, cette contrainte limite l'exploitation de l'équipement sous les réglementations WBN de l'ETSI EN 300 328 v1.8.1 à une puissance de transmission maximale (total pour toutes les chaînes de transmission après un gain d'antenne et de formation de faisceau) de 20 mW (+13 dBm) EIRP.

L'ETSI EN 300 328:2012 v1.8.1, 4.3.1.1 (FHSSM) et 4.3.2.1 (WBM) limitent la puissance d'émission maximale, après un gain d'antenne ou de formation de faisceaux à 100 mW (+20 dBm) EIRP.

L'ETSI EN 300 328:2012 v1.8.1, 4.3.1.5 (FHSSM) et 4.3.2.4 (WBM) limitent la puissance d'émission moyenne des équipements non adaptatifs et adaptatifs fonctionnant en mode non adaptatif à 10 mW (+10 dBm) EIRP. L'utilisation de l'adaptativité supprime cette restriction sur la puissance d'émission moyenne.

Lorsque la WBM sans adaptativité est revendiquée, selon l'ETSI EN 300 328:2012 v1.8.1, 4.3.2.3 chacun des répondants à une transaction D dans un intervalle n'est pas autorisé à initier une transaction D dans l'intervalle qui suit immédiatement sauf si la période de non-transmission en question satisfait à l'exigence de Tx-gap-time minimal de 3,5 ms, ce qui est supérieur de manière inhérente à Tx-sequence-time pour toute DPDU ACK/NAK venant d'être envoyée.

De façon similaire, lorsque le FHSSM sans adaptativité est revendiquée, selon l'ETSI EN 300 328:2012 v1.8.1, 4.3.1.2 chacun des répondants à une transaction D dans un intervalle n'est pas autorisé à initier une transaction D dans l'intervalle qui suit immédiatement sauf si la période de non-transmission en question satisfait à l'exigence de Tx-gap-time minimal de 5 ms, ce qui est supérieur de manière inhérente à Tx-sequence-time pour toute DPDU ACK/NAK venant d'être envoyée.

Lorsque FHSSM avec adaptivité est revendiqué, les DPDU ACK/NAK qui sont envoyées par les répondants de la transaction D en tant que réponses immédiates (dans le même intervalle) aux DPDU Data envoyées par l'initiateur de la transaction D peuvent être considérés comme signalisation SCS. Alors que LBT n'est pas exigé avant l'émission de la signalisation SCS, selon l'ETSI EN 300 328:2012 v1.8.1, 4.3.1.6.3.2, la signalisation SCS ne doit pas occuper plus de 10 % du temps de passage revendiqué. Cette restriction a un impact inverse sur la durée minimale d'intervalle pour le système en exigeant que la durée de l'intervalle soit augmentée (et le débit agrégé du système diminué en fonction) par rapport à ce qui est exigé dans d'autres cas de sorte que l'occupation de la signalisation SCS (c'est-à-dire les DPDU ACK/NAK) dans les appareils revendiquant la conformité avec FHSSM ne dépasse jamais 10 % du temps de passage nominal revendiqué.

L'approche alternative recommandée pour satisfaire aux exigences de l'ETSI EN 300 328:2012 v1.8.1, 4.3.1.6.3.2 consiste à changer dynamiquement de mode pour les répondants de la transaction D pour un fonctionnement en mode non adaptatif alors qu'ils envoient leur DPDU ACK/NAK et pendant les 5 ms qui suivent (Tx-gap-time minimal exigé), temps après lequel ils reviennent au mode de fonctionnement adaptatif. Il apparaît que la seule conséquence significative d'un tel mode de fonctionnement non adaptatif temporaire est que l'appareil répondant n'est pas autorisé à initier une transaction D dans l'intervalle de temps immédiatement consécutif sauf si la période applicable de non-émission satisfait à l'exigence de Tx-gap-time minimal.

L'ETSI EN 300 328:2012 v1.8.1, 4.3.1.3.2 (FHSSM) exige que chaque séquence cyclique à saut de voie contienne un minimum de 15 voies, qu'elles soient inactives ou actives. Selon les termes de la présente norme, cette exigence signifie que seules les entrées dlmo.Ch (Tableau 160) dont la taille a une valeur supérieure à 15 sont appropriées pour une utilisation en mode FHSSM selon la norme ETSI EN 300 328 v1.8.1. Par conséquent, lorsque des séquences à saut de voie avec des longueurs de cycles inférieures à 15 sont utilisées, le

fonctionnement selon la norme ETSI EN 300 328 v1.8.1 doit obligatoirement être conforme aux réglementations WBM de la norme EN.

L'ETSI EN 300 328 v1.8.1 n'impose aucune contrainte sur le cycle de vie de l'équipement dont la puissance de transmission maximale est ≤ 10 mW (+10 dBm) EIRP.

NOTE 6 En plus de son utilisation pendant la configuration de l'appareil, pendant laquelle la configuration souhaitée de l'utilisation finale peut être définie, cette faible puissance de transmission maximale est adéquate pour les courtes distances quand des considérations spécifiques au site ne sont pas nécessaires ou ne sont pas applicables, par exemple dans un laboratoire ou sur et entre des porte-conteneurs qui composent un train en déplacement.

Il apparaît qu'un appareil conforme à la présente norme peut être conforme aux exigences de la norme ETSI EN 300 328 v1.8.1 en étant déclaré comme fonctionnant dans l'une des six catégories et en configurant ses attributs dlmo.CountryCode (9.1.15.6), en particulier les bits 11 à 14, de manière appropriée:

1) équipement WBM de faible puissance avec dlmo.CountryCode.mode=0b"x0011x"; ou
2) équipement WBM non adaptatif avec dlmo.CountryCode.mode=0b"x0011x"; ou
3) équipement WBM adaptatif avec dlmo.CountryCode.mode=0b"x0101x"; ou
4) équipement WBM de faible puissance avec dlmo.CountryCode.mode=0b"x0011x"; ou
5) équipement FHSSM non adaptatif avec dlmo.CountryCode.mode=0b"x1001x"; ou
6) équipement FHSSM adaptatif qui change temporairement de mode pour un fonctionnement non adaptatif lorsqu'il fonctionne comme répondeur de transaction D (c'est-à-dire pour envoyer une DPDU ACK/NAK) avec dlmo.CountryCode.mode=0b"x1101x".

   NOTE 7 Bien qu'il soit possible d'utiliser un équipement FHSSM adaptatif qui ne change pas temporairement de mode, les contraintes induites par le temps de passage déclaré et la durée d'intervalle minimale exigée pour fonctionner avec cet ensemble de contraintes rendent une telle catégorie hypothétique de fonctionnement inférieure à 6) en raison du débit massivement réduit du système que des intervalles extrêmement étendus induisent nécessairement.

NOTE 8 Si les régulateurs déterminent que l'équipement conforme à la présente norme ne satisfait pas à la pleine intention de réglementation pour une ou plusieurs des six catégories possibles ci-dessus, le fonctionnement dans l'un des nœuds restants est encore possible.

Chacune des combinaisons 1) à 6) impose un ensemble différent de contraintes. Certaines sont traitées automatiquement par tous les appareils sans fil qui se conforment à la présente norme IEC, alors que d'autres dépendent du mode d'exploitation demandé. Quelle que soit le mode sélectionné et configuré par l'intermédiaire de l'attribut dlmo.CountryCode de l'appareil, l'appareil fonctionne alors de telle façon et entreprend toute action exigée pour se conformer à ces contraintes.

Pour résumer ce qui précède, les contraintes réglementaires qui exigent une autosurveillance sont:

a) pour un fonctionnement dans les modes 1 et 4, limiter la puissance de sortie maximale de l'émetteur, P_{out}Max, à moins de 10 mW (+10 dBm) EIRP;
b) pour un fonctionnement dans les modes 2 et 3, limiter la puissance de sortie maximale, P_{out}Max, à 20 mW (+13 dBm) EIRP, ce qui représente une densité de 10 mW/MHz EIRP pour la signalisation de IEEE 802.15.4 2.4 GHz DSSS;
c) pour un fonctionnement dans les modes 5 et 6, limiter la puissance de sortie maximale de l'émetteur, P_{out}Max, à 100 mW (+20 dBm) EIRP;
d) pour un fonctionnement dans le mode 2, limiter le nombre total d'émissions, des DPDU Data et des DPDU ACK/NAK, de telle sorte que la puissance de sortie moyenne de l'émetteur, P_{out}Avg, est 10 mW (+10 dBm) EIRP ou moins sur tous les intervalles de mesure de 1,0 s;
e) pour un fonctionnement dans le mode 5, limiter le nombre total d'émissions, des DPDU Data et des DPDU ACK/NAK, de telle sorte que la puissance de sortie moyenne de

l'émetteur, $P_{out}Avg$, est 10 mW (+10 dBm) EIRP ou moins sur chaque intervalle de mesure de 100 fois la durée d'un intervalle de temps;

f) pour un fonctionnement dans le mode 6, limiter le nombre total d'émissions des ACK/NAK DPDU, de telle sorte que la puissance de sortie moyenne de l'émetteur, $P_{out}Avg$, utilisé pendant la transmission des ACK/NAK DPDU est 10 mW (+10 dBm) EIRP ou moins sur chaque intervalle de mesure de 100 fois la durée d'un intervalle de temps;

NOTE 9 La majorité de l'occupation de voie par des appareils fonctionnant en mode 6) se produit lors de l'envoi de DPDU Data. De telles émissions sont considérées comme de l'adaptivité selon l'ETSI EN 300 328:2012 v1.8.1, 4.3.1.5; donc, la contrainte d) ne s'applique pas à elles. Cependant, des appareils fonctionnant en catégorie 6) émettent des DPDU ACK/NAK en mode non adaptatif pour lesquelles la contrainte d) s'applique conformément à l'ETSI EN 300 328:2012 v1.8.1, 4.3.1.5. Il apparaît par conséquent que seule la totalité de telles DPDU ACK/NAK émises par un appareil fonctionnant en mode 6) est sujette à la limitation de puissance de la contrainte d). Si cette interprétation de la norme ETSI EN 300 328 v1.8.1 est correcte, alors la contrainte d) affectera principalement les routeurs dorsaux (BBR), qui, au sein d'une automatisation WISN, sont essentiellement des récepteurs de publications de valeurs et statuts de traitement et de rapports d'alertes émanant des appareils WISN de terrain. Les BBR agissant comme répondeurs de transaction dans le cas d'une transaction en duodiffusion peuvent être davantage impactés que ceux qui ne reçoivent la duodiffusion.

g) pour un fonctionnement en mode 2 et 5, satisfaire à l'exigence d'intervalle Tx-gap-time de non-émission après l'émission d'une DPDU Data;

NOTE 10 Cette contrainte est satisfaite automatiquement lorsque la durée est ≥ 8,508 ms en mode 2, ou ≥ 9,256 ms en mode 5).

h) pour un fonctionnement en mode 2, 5 et 6, satisfaire à l'exigence d'intervalle Tx-gap-time de non-émission après l'émission d'une DPDU ACK/NAK.

Pour d), e), f), g) et h), l'équipement contrôle dynamiquement son activité récente pour éviter d'émettre lorsque cela constitue une violation de l'une de ces contraintes.

NOTE 11 Alors qu'un gestionnaire de système peut programmer l'activité d'appareil pour satisfaire aux exigences de d), e), f), g) et h), il est de la propre responsabilité de l'appareil de surveiller son activité récente et d'empêcher l'émission lorsque celle-ci constituerait une violation des contraintes de réglementation. Ainsi, la responsabilité ultime pour le fonctionnement d'un ensemble d'appareils repose sur les appareils individuels eux-mêmes, et non sur quelque gestionnaire distant qui pourrait être corrompu par une cyberattaque réussie sur un seul appareil.

Bibliographie

IEC 61158 (toutes les parties), *Réseaux de communication industriels – Spécifications des bus de terrain*

IEC 61499-4:2013, *Blocs fonctionnels – Partie 4: Règles pour les profils de conformité*

IEC 61512-1, *Contrôle-commande des processus de fabrication par lots - Partie 1: Modèles et terminologie*

IEC 61804-3, *Blocs fonctionnels (FB) pour les procédés industriels – Partie 3: Langage de description électronique de produit (EDDL)*

IEC 62264-1, *Intégration des systèmes entreprise-contrôle – Partie 1: Modèles et* terminologie

IEC/TS 62351-2:2008, *Power systems management and associated information exchange – Data and communications security – Part 2: Glossary of terms* (disponible en anglais seulement)

IEC/TR 62390:2005, *Common automation device – Profile guideline* (disponible en anglais seulement)

IEC/TS 62443-1-1:2009, *Industrial communication networks – Network and system security – Part 1-1: Terminology, concepts and models* (disponible en anglais seulement)

IEC 62591, *Réseaux de communications industriels – Réseau de communications sans fil et profils de communication – WirelessHART™*

IEC 62601, *Industrial communication networks – Fieldbus specifications – WIA-PA communication network and communication profile* (disponible en anglais seulement)

IEC 62657-2, *Réseaux de communication industriels – Réseaux de communication sans fil - Partie 2: Gestion de coexistence*

ISO/IEC 2375, *Technologies de l'information — Procédure pour l'enregistrement des séquences d'échappement et des jeux de caractères codés*

ISO/IEC 2382-14:1997, *Technologies de l'information – Vocabulaire – Partie 14: Fiabilité, maintenabilité et disponibilité*

ISO/IEC 7498-1:1994 telle que corrigée et réimprimée en 1996, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Modèle de référence de base:_ Le modèle de base*

ISO/IEC 7498-2, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Modèle de référence de base – Partie 2: Architecture de sécurité*

ISO/IEC 7498-3:1997, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Modèle de référence de base: Dénomination et addressage*

ISO/IEC 7498-4 *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Modèle de référence de base – Partie 4: Cadre général de gestion*

ISO/IEC 9646-7, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Essais de conformité – Méthodologie générale et procédures – Partie 7: Déclarations de conformité des mises en oeuvre*

ISO/IEC 9796-2:2010, *Technologies de l'information — Techniques de sécurité — Schémas de signature numérique rétablissant le message — Partie 2: Mécanismes basés sur une factorisation entière*

ISO/IEC 9797-1:2011, *Technologies de l'information — Techniques de sécurité — Codes d'authentification de message (MAC) — Partie 1: Mécanismes utilisant un chiffrement par blocs*

ISO/IEC 9797-2:2011, *Technologies de l'information — Techniques de sécurité — Codes d'authentification de message (MAC) – Partie 2: Mécanismes utilisant une fonction de hachage dédiée*

ISO/IEC 9798-1:2010, *Technologies de l'information — Techniques de sécurité — Authentification d'entité — Partie 1: Généralités*

ISO/IEC 10116:2006, *Technologies de l'information — Techniques de sécurité — Modes opératoires pour un chiffrement par blocs de n-bits*

ISO/IEC 10118-2, *Technologies de l'information — Techniques de sécurité — Fonctions de brouillage — Partie 2: Fonctions de brouillage utilisant un chiffrement par blocs de n bits*

ISO/IEC 10118-3, *Technologies de l'information — Techniques de sécurité — Fonctions de brouillage — Partie 3: Fonctions de brouillage dédiées*

ISO/IEC 10181-1:1996, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Cadres de sécurité pour les systèmes ouverts: Aperçu*

ISO/IEC 11770-1:2010, *Technologies de l'information — Techniques de sécurité — Gestion de clés — Partie 1: Cadre général*

ISO/IEC 11770-2, *Technologies de l'information — Techniques de sécurité — Gestion de clés — Partie 2: Mécanismes utilisant des techniques symétriques*

ISO/IEC 11770-3:2008 *Technologies de l'information — Techniques de sécurité — Gestion de clés — Partie 3: Mécanismes utilisant des techniques asymétriques*

ISO/IEC 15408 (toutes les parties), *Technologies de l'information – Techniques de sécurité – Critères d'évaluation pour la sécurité TI*

ISO/IEC 18028-3:2005, *Technologies de l'information — Techniques de sécurité — Sécurité de réseaux TI — Partie 3: Communications de sécurité entre réseaux utilisant des portails de sécurité*

ISO/IEC 18031:2011, *Technologies de l'information — Techniques de sécurité – Génération de bits aléatoires*

ISO/IEC 18033-1:2005, *Technologies de l'information — Techniques de sécurité — Algorithmes de chiffrement — Partie 1: Généralités*

ISO/IEC 18033-2, *Technologies de l'information — Techniques de sécurité — Algorithmes de chiffrement — Partie 2: Chiffres asymétriques*

ISO/IEC 19790:2012, *Technologies de l'information — Techniques de sécurité — Exigences de sécurité pour les modules cryptographiques*

ISO/IEC 26907:2009, *Technologies de l'information — Téléinformatique — Norme PHY et MAC à bande ultralarge et haut débit*

ISO/IEC 27000:2014, *Technologies de l'information— Techniques de sécurité — Systèmes de management de la sécurité de l'information — Vue d'ensemble et vocabulaire*

ISO/IEC/IEEE 60559, *Information technology – Microprocessor Systems – Floating-Point arithmetic*

ISO 2382-12:1988, *Systèmes de traitement de l'information – Vocabulaire – Part 12: Périphériques*

ISO 3166-1, *Codes pour la représentation des noms de pays et de leurs subdivisions — Partie 1:Codes de pays*

ISO 11568-2:2012, *Services financiers— Gestion de clés (services aux particuliers) — Partie 2: Algorithmes cryptographiques symétriques, leur gestion de clés et leur cycle de vie*

ISO 11568-4:2007, *Services financiers— Gestion de clés (services aux particuliers) — Partie 4: Cryptosystèmes asymétriques — Gestion des clés et cycle de vie*

ISO 18435-2:2012, *Systèmes d'automatisation industrielle et intégration — Diagnostics, évaluation des moyens et intégration des applications de maintenance — Partie 2: Descriptions et définitions des éléments de matrice du domaine d'application*

ISO 21188:2006, *Infrastructure de clé publique pour services financiers — Pratique et cadre politique*

IEEE 802.1Q, *IEEE Standard for Local and metropolitan area networks – Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks* (disponible en anglais seulement)

IEEE 802.3, *IEEE Standard for Information technology-Specific requirements – Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications* (disponible en anglais seulement)

NOTE 1 ISO/IEC 8802-3 is based on IEEE 802.3, usually with some delay in publication.

IEEE 802.11:2012, *IEEE standards for information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 11: Wireless LAN medium access control (MAC) and physical layer (PhL) specifications* (disponible en anglais seulement)

NOTE 2 ISO/IEC 8802-11 is based on IEEE 802.11:2012, usually with some delay in publication.

IEEE 802.15.1:2005, *IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements. Part 15.1: Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs)* (disponible en anglais seulement)

NOTE 3 ISO/IEC 8802-15-1 is based on IEEE 802.15.1:2005, usually with some delay in publication.

IEEE Std 802.15.4e-2012, (Amendment to IEEE Std 802.15.4-2011), *IEEE Standard for Local and metropolitan area networks- Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer* (disponible en anglais seulement)

IEEE 802.16:2012, *IEEE Standard for local and metropolitan area networks–Part 16: Air interface for fixed broadband wireless access systems* (disponible en anglais seulement)

IERS conventions: IERS technical note 32 (disponible en anglais seulement)

IETF RFC 1350, *The TFTP protocol, rev. 2* (disponible en anglais seulement)

IETF RFC 2347, *TFTP option extension* (disponible en anglais seulement)

IETF RFC 2348, *TFTP blocksize option* (disponible en anglais seulement)

IETF RFC 2349, *TFTP timeout interval and transfer size options* (disponible en anglais seulement)

IETF RFC 2525, *Known TCP implementation problems* (disponible en anglais seulement)

IETF RFC 3280, *Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile* (disponible en anglais seulement)

IETF RFC 5348, *TCP-friendly rate control (TFRC): Protocol specification* (disponible en anglais seulement)

IETF RFC 4949, *Internet security glossary, rev. 2* (disponible en anglais seulement)

ERC Recommendation 70-03, *Relating to the use of Short Range Devices (SRD), Annex 1 Band h and Annex 3 Band a* (disponible en anglais seulement)

ETSI EN 300 328 1 v1.8.1, *Electromagnetic compatibility and Radio spectrum Matters (ERM); Wideband transmission systems; Data transmission equipment operating in the 2,4 GHz ISM band and using wide band modulation techniques; Harmonized EN covering the essential requirements of article 3.2 of the R&TTE Directive* (disponible en anglais seulement)

ETSI EN 300 440 1 v1.6.1, *Electromagnetic compatibility and Radio spectrum Matters (ERM); Short range devices; Radio equipment to be used in the 1 GHz to 40 GHz frequency range; Part 1: Technical characteristics and test methods* (disponible en anglais seulement)

ETSI EN 300 440 2 v1.4.1, *Electromagnetic compatibility and Radio spectrum Matters (ERM); Short range devices; Radio equipment to be used in the 1 GHz to 40 GHz frequency range; Part 2: Harmonized EN covering essential requirements of article 3.2 of the R&TTE Directive* (disponible en anglais seulement)

ISC RSS 210, *Radio standards specification 210 – Low-power license-exempt radio communication devices (all frequency bands): Category I equipment* (disponible en anglais seulement)

ANSI X9.82-1, *Random number generation – Part 1: Overview and basic principles* (disponible en anglais seulement)

ISA Handbook of Measurement Equations and Tables, 2nd Edition, ISBN 978-1-55617-946-4 (disponible en anglais seulement) (disponible en anglais seulement)

ISA 100.11a, *Wireless Systems for Industrial Automation: Process Control and Related Applications* (disponible en anglais seulement)

ISA TR100.00.01-2006, *The Automation Engineer's Guide to Wireless Technology Part 1 – The Physics of Radio, a Tutorial* (disponible en anglais seulement)

[US] FIPS 186-3, *Digital Signature Standard (DSS)* (disponible en anglais seulement)

[US] FIPS 197, *Advanced encryption standard (AES)* (disponible en anglais seulement)

[US] FIPS 198, *The keyed-hash message authentication code (HMAC)* (disponible en anglais seulement)

[US] NIST SP 800-22, *A statistical test suite for random and pseudorandom number generators for cryptographic applications* (disponible en anglais seulement)

[US] NIST SP 800-38C, *Recommendation for block cipher modes of operation – The CCM mode for authentication confidentiality* (disponible en anglais seulement)

[US] NIST SP 800-56A, *Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography* (disponible en anglais seulement)

[US] NIST SP 800-57, *Recommendation for key management – Part 1: General* (disponible en anglais seulement)

[US] NIST SP 800-57, *Recommendation for key management – Part 2: Best practices for key management organization* (disponible en anglais seulement)

[US] NIST SP 800-88:2012, rev. 1, *Guidelines for media sanitization* (disponible en anglais seulement)

[US] Code of Federal Regulations (CFR) Title 47, *Chapter I, Part 15 – Telecommunication – Part 15: Radio frequency devices* (disponible en anglais seulement)

NAMUR Recommendation NE105, *Specifications for integrating fieldbus devices* (disponible en anglais seulement)

NAMUR Recommendation NE107, *Self-monitoring and diagnostics of field devices* (disponible en anglais seulement)

Guidelines for 64-bit Global Identifier (EUI-64™), available at http://standards.ieee.org/develop/regauth/tut/eui64.pdf. (disponible en anglais seulement)

HCF_SPEC-183, *Common Tables Specification*, available to members of the HART Communication Foundation, http://www.hartcomm.org (disponible en anglais seulement)

A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, *Handbook of applied cryptography*, ISBN 0-8493-8523-7, 1996 (disponible en anglais seulement)

D. R. L. Brown, R. P. Gallant, S. A. Vanstone, *Provably secure implicit certificate schemes*, pp. 156-165 of ISBN 3-540-44079-8 (disponible en anglais seulement)

F. Stajano, *The resurrecting duckling: What next?*, in *Proceedings of the 8th international workshop on security protocols*, B. Crispo, M. Roe, and B. Crispo, Eds., Lecture notes in computer science, Vol. 2133, Berlin: Springer-Verlag, April 2000 (disponible en anglais seulement)

F. Stajano, R. Anderson, *The resurrecting duckling: Security issues in ad-hoc wireless networks*, in Proceedings of the 7th international workshop on security protocols, B. Christianson, B. Crispo, J.A. Malcolm, and M. Roe, Eds., Lecture notes in computer science, Vol. 1796, Berlin: Springer-Verlag, 1999 (disponible en anglais seulement)

J. Jonsson, *On the security of CTR + CBC-MAC*, in *Proceedings of selected areas in cryptography – SAC 2002*, K. Nyberg, H. Heys, Eds. Lecture notes in computer science, Vol. 2595, pp. 76-93, Berlin: Springer, 2002 (disponible en anglais seulement)

J. Jonsson, *On the security of CTR + CBC-MAC, NIST mode of operation – Additional CCM documentation*, http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/ccm/ccm-ad1.pdf (disponible en anglais seulement)

PKIX, L. Bassham, R. Housley, W. Polk, *Algorithms and identifiers for the internet X.509 Public key infrastructure certificate and CRL profile*, ftp://ftp.isi.edu/in-notes/rfc3279.txt (disponible en anglais seulement)

P. Rogaway, D. Wagner, *A critique of CCM*, IACR ePrint Archive 2003-070, April 13, 2003 (disponible en anglais seulement)

R. Housley, D. Whiting, N. Ferguson, *Counter with CBC-MAC (CCM)*, submitted to NIST., June 3, 2002 (disponible en anglais seulement)

———————

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

3, rue de Varembé
PO Box 131
CH-1211 Geneva 20
Switzerland

Tel: + 41 22 919 02 11
Fax: + 41 22 919 03 00
info@iec.ch
www.iec.ch