IEC PAS 62948

Edition 1.0 2015-04

PUBLICLY AVAILABLE SPECIFICATION

PRE-STANDARD

colour inside

Industrial networks – Wireless communication network and communication profiles – WIA-FA

IEC PAS 62948:2015-04(en)

IEC PAS 62948

Edition 2.0 2015-04

PUBLICLY AVAILABLE SPECIFICATION

PRE-STANDARD

colour inside

Industrial networks – Wireless communication network and communication profiles – WIA-FA

INTERNATIONAL ELECTROTECHNICAL COMMISSION

ICS 25.040.40; 35.100.01

ISBN 978-2-8322-2477-9

Warning! Make sure that you obtained this publication from an authorized distributor.

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

____

INDUSTRIAL NETWORKS – WIRELESS COMMUNICATION NETWORK AND COMMUNICATION PROFILES – WIA-FA

FOREWORD

A PAS is a technical specification not fulfilling the requirements for a standard, but made available to the public.

IEC PAS 62948 has been processed by subcommittee 65C: Industrial networks, of IEC technical committee 65: Industrial-process measurement, control and automation.

The text of this PAS is based on the following document:	This PAS was approved for publication by the P-members of the committee concerned as indicated in the following document
Draft PAS	**Report on voting**
65C/784/PAS	65C/789/RVD

Following publication of this PAS, which is a pre-standard publication, the technical committee or subcommittee concerned may transform it into an International Standard.

This PAS shall remain valid for an initial maximum period of 3 years starting from the publication date. The validity may be extended for a single period up to a maximum of 3 years, at the end of which it shall be published as another type of normative document, or shall be withdrawn.

A bilingual version of this publication may be issued at a later date.

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.

INDUSTRIAL NETWORKS – WIRELESS COMMUNICATION NETWORK AND COMMUNICATION PROFILES – WIA-FA

1 Scope

This PAS specifies the system architecture and communication protocol of WIA-FA (Wireless Networks for Industrial Automation – Factory Automation) based on IEEE STD 802.11-2012 Physical Layer (PHY).

This PAS applies to wireless network systems for factory automation measuring, monitoring and control.

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.

IEC 61588, *Precision clock synchronization protocol for networked measurement and control systems*

IEC 61499, *The industrial process measurement and control system function blocks*

ISO/IEC 7498-1, *Information Technology – Open Systems Interconnection – Basic Reference Model – The Basic model*

IEEE STD 802.11-2012, *IEEE Standard 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 (PHY) specifications*

3 Terms, definitions, abbreviations, and conventions

3.1 Terms and definitions

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

3.1.1
absolute timeslot number
number of timeslots from the start of the network, generally denoting the current timeslot

Note 1 to entry: The value is incremented by one, usually the same with the current timeslot sequence number. Once the maximum value (2^{48}—1) is reached, the value is reset to 0.

3.1.2
access device
device installed in the field, which is responsible for forwarding the sensor data, alarm and network management related information of the field device to the gateway device, or forwarding control signals, management information and configuration information of the gateway device to field devices

3.1.3
aggregation
process of aggregating data from multiple user application objects into one packet, or aggregating several frames into one frame

3.1.4
application configuration
configuration process for user application processes in field devices to finish certain jobs during the application of factory automation

3.1.5
application sublayer
protocol sublayer that provides data and management services for the application layer

3.1.6
backoff
process of retrying a frame in pre-determined retransmission timeslots if the first transmission is failed

3.1.7
beacon
special frame broadcast by the access device in the WIA-FA network

Note 1 to entry: To join the WIA-FA network, a new field device should first listen to beacons.

3.1.8
channel
RF medium used to convey a frame from a sender to a receiver

3.1.9
coexistence
state in which all wireless communication solutions of a plant using shared medium fulfil all their application communication requirements

[SOURCE: IEC 62657-2:2013, 3.1.12]

3.1.10
communication resource
channels and timeslots used to transport a frame

3.1.11
disaggregation
process of dividing the aggregated packet into data of multiple user application objects, or dividing the aggregated frame into multiple frames

3.1.12
field device
device installed in industrial field and connected by sensors and actuators, which is used for transmitting field data and receiving control commands

3.1.13
gateway device
device that connects the WIA-FA network to other plant networks

3.1.14
handheld device
portable device used for network provisioning and firmware updating

3.1.15
heartbeat message
signal sent by the working gateway device to the redundant ones, which indicates that the gateway device is working properly

3.1.16
host computer
computer through which operation, maintenance and management personnel interact with the WIA-FA network

Note 1 to entry: Host computer performs functions of configuration, provisioning, and data display.

3.1.17
interoperability
ability of two or more network systems to exchange information and to make mutual use of the information that has been exchanged

[SOURCE: ISO/IEC TR 10000-1:1998, 3.2.1 – modified by replacing "IT systems" with "network systems"]

3.1.18
joining
process in which a WIA-FA device is authenticated and allowed to participate in the WIA-FA network

3.1.19
link
interconnecting path between two neighbouring devices, which consists of a set of parameters

Note 1 to entry: A set of parameters include link identifier, link type, destination address, relative timeslot number, current channel index, and superframe identifier.

Note 2 to entry: A WIA-FA network has only one primary gateway device.

3.1.20
network address
8-bit or 16-bit unsigned integer uniquely identifying the device in the WIA-FA network

Note 1 to entry: It is also called short address.

3.1.21
network configuration
process of configuring parameters for WIA-FA devices to maintain network operation and communication

3.1.22
network manager
logical role responsible for configuring the network, allocating communication resources, monitoring and reporting network performances

Note 1 to entry: There is only one network manager in a WIA-FA network.

3.1.23
passive leaving
process in which an online field device is instructed by the gateway device to leave the WIA-FA network

3.1.24
physical address
EUI-64 bits uniquely identifying the device in the WIA-FA network

Note 1 to entry: : It is also called long address.

Note 2 to entry: : A physical address is assigned by a manufacturer.

3.1.25
primary gateway device
gateway device working in a WIA-FA network

3.1.26
provisioning
process of pre-configuring some static information that includes network identifier, security level, join key, and shared key

3.1.27
redundant gateway device
hot backup of the primary gateway device

3.1.28
relative timeslot number
timeslot number counted from the start of a superframe

3.1.29
security manager
logical role responsible for configuring the security policies of the whole network, managing keys, and authenticating devices

3.1.30
superframe
collection of channels and timeslots repeatedly appear in one cycle

Note 1 to entry: It specifies the transmitting or receiving time for periodic communication.

3.1.31
timeslot
basic time unit used for data exchange in WIA-FA network

Note 1 to entry: Its duration is configurable in the WIA-FA network.

3.2 Abbreviations

For the purposes of this document, the following abbreviations apply.

ACK	Acknowledgement
AI	Analog Input
AD	Access Device
AL	Application Layer
ASLDE	Application SubLayer Digital Entity
ASLM	ASL State Machine
AMCTL	ASL State Machine of Client
AMSV	ASL State Machine of Server
AMPB	ASL State Machine of Publisher
AMSB	ASL State Machine of Subscriber
AMRS	ASL State Machine of Report Source
AMRK	ASL State Machine of Report Sink

AO	Analog Output
ASL	Application Sub-layer
ASN	Absolute Slot Number
APDU	Application Protocol Data Unit
ASDU	Application Service Data Unit
C/S	Client/Serve
CCM*	Extension of counter with cipher block chaining message authentication code
DI	Digital Input
DO	Digital Output
DGO	DisaGgregation Object
DLL	Data Link Layer
DLPDU	Data link Layer Protocol Data Unit
DMAP	Device Management Application Process
EIRP	Equivalent Isotropic Radiated Power
ENC	ENCryption
FCS	Frame Check Sequence
FD	Field Device
FDMA	Frequency Division Multiple Access
GW	Gateway Device
HC	Host Computer
HD	Handheld Device
HMAC	keyed-Hash Message Authentication Code
ID	Identifier
KED	Data Encryption Key
KEDB	Broadcast Data Encryption Key
KEDU	Unicast Data Encryption Key
KEK	Key Encryption Key
KJ	Join Key
KS	Shared Key
LSB	Least Significant Bit
LQI	Link Quality Indication
MAC	Medium Access Control
MIB	Management Information Base
MIC	Message Integrity Code
MSB	Most Significant Bit
NACK	Negative Acknowledgement
NRT	Non-Real-Time
NONCE	Number used once, a value that has (at most) a negligible chance of repeating
PAGO	Packet Aggregation Object
PDU	Protocol Data Unit

PHY	PHYsical layer
P/S	Publish/Subscriber
R/S	Report/Sink
SAP	Service Access Point
SM	Security Manager
TDMA	Time Division Multiple Access
UAO	User Application Object
UAP	User Application Process
VCR	Virtual Communication Relationship
WIA-FA	Wireless Network for Industrial Automation – Factory Automation

3.3 Conventions

For the purposes of the state machines and state transitions in this PAS, the following conventions apply.

This PAS uses a diagram to represent a state machine. The conventions used in the state machines are shown in Figure 1.

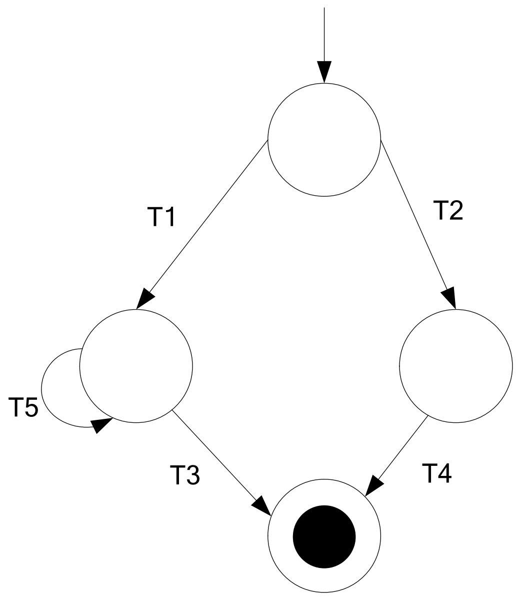

Figure 1 – Conventions used for state machines

The conventions used in the state machines are as follows:

- The labeled circles represent states that a device can be in.
- State transitions are directed lines. They show which state a component leaves and which state it transitions to.
- A transition is labeled by T with the events that caused it and the corresponding actions (this may be empty). The events (above the line) causing the transition is separated from the resulting actions (below the line) by a horizontal line.
- || Logical "OR".

The conventions used in the state transitions are shown in Table 1.

Table 1 – Conventions used for state transitions

#	Current state	Event/conditions =>action	Next state
Name of transition	The current state to which this state transition applies	Events or conditions that trigger this state transaction. => The actions that are taken when the above events or conditions are met. The actions are always indented below events or conditions	The next state after the actions in this transition is taken

The conventions used in the state transitions are as follows:

– = A logical condition to indicate an item on the left is equal to an item on the right.

– := Value of an item on the left is replaced by value of an item on the right. If an item on the right is a parameter, it comes from the primitive shown as an input event.

– == A logical condition to judge an item on the left is equal to an item on the right.

– && Logical "AND".

– || Logical "OR".

– != A logical condition to indicate an item on the left is not equal to an item on the right.

This construct allows the execution of a sequence of actions in a loop within one transition.

The loop is executed for all values from start_value to end_value.

Example:
```
for (Identifier = start_value to end_value)
        actions
end
```

This construct allows the execution of alternative actions depending on some condition (which

may be the value of some identifier or the outcome of a previous action) within one

transition.

Example:
```
If (condition)
    actions
else
    actions
endif
```

4 Data coding

4.1 Overview

WIA-FA data coding specifies the machine independent syntax for the data conveyed by each layer services. WIA-FA supports the definition and transfer of both basic and struct data types.

Basic types are atomic types that cannot be decomposed into more elemental types. Struct types are types composed of basic types and other struct types. Their complexity and depth of nesting is not constrained by this PAS.

4.2 Basic data type coding

4.2.1 Integer coding

The integer is a signed value, as shown in Figure 2 and Table 2. For this data type, MSB is transmitted first, then the subsequent octets, and finally the LSB.

Notations: Integer8, Integer16, Integer24, Integer32		
Data type	**Value range**	**Length**
Integer8	$-128 \le i \le 127$	One octet
Integer16	$-32\,768 \le i \le 32\,767$	Two octets
Integer24	$-2^{23} \le i \le 2^{23} - 1$	Three octets
Integer32	$-2^{31} \le i \le 2^{31} - 1$	Four octets

Complement binary notation
MSB is the bit followed by the first octet (SN)
SN = 0: positive and 0
SN = 1: negative		

Figure 2 – Integer coding

Table 2 – Definition of integer data type

octet	**bit**							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	SN	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}

4.2.2 Unsigned coding

The unsigned value is coded as shown in Figure 3 and Table 3. For this data type, MSB is sent first, then the subsequent octets, and finally the LSB.

Notations: Unsigned8, Unsigned16, Unsigned24, Unsigned32, Unsigned40, Unsigned48, Unsigned64, Unsigned80		
Data type	**Value range**	**Length**
Unsigned8	$0 \le i \le 255$	One octet
Unsigned16	$0 \le i \le 65\,535$	Two octets
Unsigned24	$0 \le i \le 2^{24} - 1$	Three octets
Unsigned32	$0 \le i \le 2^{32} - 1$	Four octets
Unsigned40	$0 \le i \le 2^{40} - 1$	Five octets
Unsigned48	$0 \le i \le 2^{48} - 1$	Six octets
Unsigned64	$0 \le i \le 2^{64} - 1$	Eight octets
Unsigned80	$0 \le i \le 2^{80} - 1$	Ten octets

Figure 3 – Unsigned coding

Table 3 – Unsigned16 coding

octet	**bit**							
	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}

4.2.3 Float coding

The value of floating point is coded as shown in Figure 4 and Figure 5.

The MSB of the sign and the exponent is sent first, and then the remained bits of the exponent and the bits from MSB to LSB of the fraction. If the value of the floating point data is unknown, 0x7F and 0xA0 are firstly sent, followed by 0x00, which means “Not-a-number”.

Notations:	**Single Float (four octets)**
Value range	See in IEC 60559, Short Real Number (32 bits)
coding	See in IEC 60559, Short Real Number (32 bits)

	MSB							LSB
Bit	7	6	5	4	3	2	1	0
Octet	**Exponent (E)**							
1	SN	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}
	Fraction (F)							
2	2^{0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
	Fraction (F)							
3	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}
	Fraction (F)							
4	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}	2^{-21}	2^{-22}	2^{-23}

SN: sign 0 = positive, 1 = negative

Figure 4 – Single float coding

Notations:	**Double Float (eight octets)**
Value range	See in IEC 60559, Short Real Number, 64 bits
coding	See in IEC 60559, Short Real Number, 64 bits

	MSB							LSB
bit	7	6	5	4	3	2	1	0
octet	**Exponent (E)**							
1	SN	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}
	Exponent (E)				**Fraction (F)**			
2	2^{3}	2^{2}	2^{1}	2^{0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	Fraction (F)							
3	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}
	Fraction (F)							
4	2^{-13}	2^{-14}	2^{-15}	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}
	Fraction (F)							
5	2^{-21}	2^{-22}	2^{-23}	2^{-24}	2^{-25}	2^{-26}	2^{-27}	2^{-28}
	Fraction (F)							
6	2^{-29}	2^{-30}	2^{-31}	2^{-32}	2^{-33}	2^{-34}	2^{-35}	2^{-36}
	Fraction (F)							
7	2^{-37}	2^{-38}	2^{-39}	2^{-40}	2^{-41}	2^{-42}	2^{-43}	2^{-44}
	Fraction (F)							
8	2^{-45}	2^{-46}	2^{-47}	2^{-48}	2^{-49}	2^{-50}	2^{-51}	2^{-52}

SN: sign 0 =positive, 1=negative

Figure 5 – Double float coding

4.2.4 Octetstring coding

The Octetstring coding manner is shown Table 4. For the data with N octets, the MSB of the most significant octet of PDU is first sent.

Table 4 – Octetstring coding

octet	bit							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	2^{8N-1}	2^{8N-2}	2^{8N-3}	2^{8N-4}	2^{8N-5}	2^{8N-6}	2^{8N-7}	2^{8N-8}
2	2^{8N-9}	2^{8N-10}	2^{8N-11}	2^{8N-12}	2^{8N-13}	2^{8N-14}	2^{8N-15}	2^{8N-16}
…	…	…	…	…	…	…	…	…
N	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

4.2.5 Bit Field coding

Bit Field data are used to encode objects as single-bit data. The bit coding manner is shown in Table 5, Table 6, and Table 7. This data type is defined as a series of eight bits. For the data with two octets, the most significant octet of PDU is sent first. For each bit of the bit field data type, there is a corresponding definition table.

Table 5 – Coding of Bit Field data with one octet

octet	bit							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	7	6	5	4	3	2	1	0

Table 6 – Coding of Bit Field data with two octets

octet	bit							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	15	14	13	12	11	10	9	8
2	7	6	5	4	3	2	1	0

Table 7 – Coding of Bit Field data with three octet

octet	bit							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	23	22	21	20	19	18	17	16
2	15	14	13	12	11	10	9	8
3	7	6	5	4	3	2	1	0

4.2.6 TimeData coding

This data type is an unsigned integer with a length of 64 bits, indicating time incremented by 1us.

4.2.7 KeyData coding

This data type is an unsigned integer with a length of 128 bits.

4.3 Structured data type coding

4.3.1 Structure type coding

A structure is made of an ordered set of heterogeneously different typed data called members, which may be basic data type or structured data type. The member of a structure is identified by MemberID. The structure data may be accessed as a whole, or just one of its members may be accessed by specifying the MemberID.

4.3.2 List type coding

A list is composed of an ordered set of homogeneously same typed elements called records, which may be basic data type or structured data type. The record of a list is identified by FirstStoreIndex. The list data may be accessed as a whole, or one or several of its records may be accessed by specifying the starting FirstStoreIndex and the count.

5 WIA-FA overview

5.1 Device types

The document specifies five types of WIA-FA devices:

host computer (HC);

gateway device (GW);

access device (AD);

field device (FD);

handheld device (HD).

To improve availability and reliability, WIA-FA network allows the existence of redundant gateway device as a hot backup, multiple access devices working in parallel.

NOTE 1 In the WIA-FA network, the redundant gateway device (see 3.1.27 and the primary gateway device (see 3.1.25) use the same network address. They adopt wired connection and synchronous update. The redundant gateway device periodically receives the heartbeat signal from the primary gateway device in wired connection manner. If the redundant gateway device does not receive the heartbeat signal in the period of PriGwFailureTime (see Table 14), it will take over all works of the original primary gateway device.

NOTE 2 Access devices connect the gateway device in the wired manner (see Clause 9). Multiple access devices concurrently work.

5.1.1 Host computer

Host computer is the interface for the operation, maintenance and management personnel to execute functions of application configuration, network configuration, and data display. Its specific implementation is beyond the scope of this PAS.

5.1.2 Gateway device

The main functions of the gateway device are listed as follows:

– Providing interconnection interfaces between WIA-FA network and other external networks, such as fieldbus, using data mapping and protocol conversion means;
– Responsible for network management and security management;
– Communicating with other WIA-FA devices through access devices, and exchanging information between devices;
– Acting as the unique clock source in the WIA-FA network for network time synchronization.

5.1.3 Access device

The main functions of the access device are listed as follows:

– Receiving data gathered by field device and forwarding them to the gateway device;
– Forwarding the control command of gateway device to the actuators of field device;
– Forwarding the management information and configuration information of gateway device to the field devices;
– Receiving alarms and network management information of field device and forwarding them to the gateway device.

NOTE The access devices and gateway devices are connected by wire. Their synchronization method is beyond the scope of this PAS.

5.1.4 Field device

Field devices are installed in industrial field and connect sensors and actuators. Field devices collect field application data and control production process. The power supply modes of field devices include wired power supply, battery power supply, etc., which are beyond the scope of this PAS.

5.1.5 Handheld device

Handheld devices are portable devices that used for provisioning field device, access device, and gateway device. A handheld device communicates with its direct connected device, and does not communicate with other WIA-FA devices. Handheld devices use RS-232 maintenance port to provision field device, access device and gateway device and write security level, join keys, shared keys(except security level 0), and network Identifier (ID) for them.

5.2 Network topology

As shown in Figure 6, WIA-FA supports the enhanced star topology, which is comprised of a center and a number of field devices. The center is comprised of one gateway device (redundant gateway device can exist) and one or several access devices.

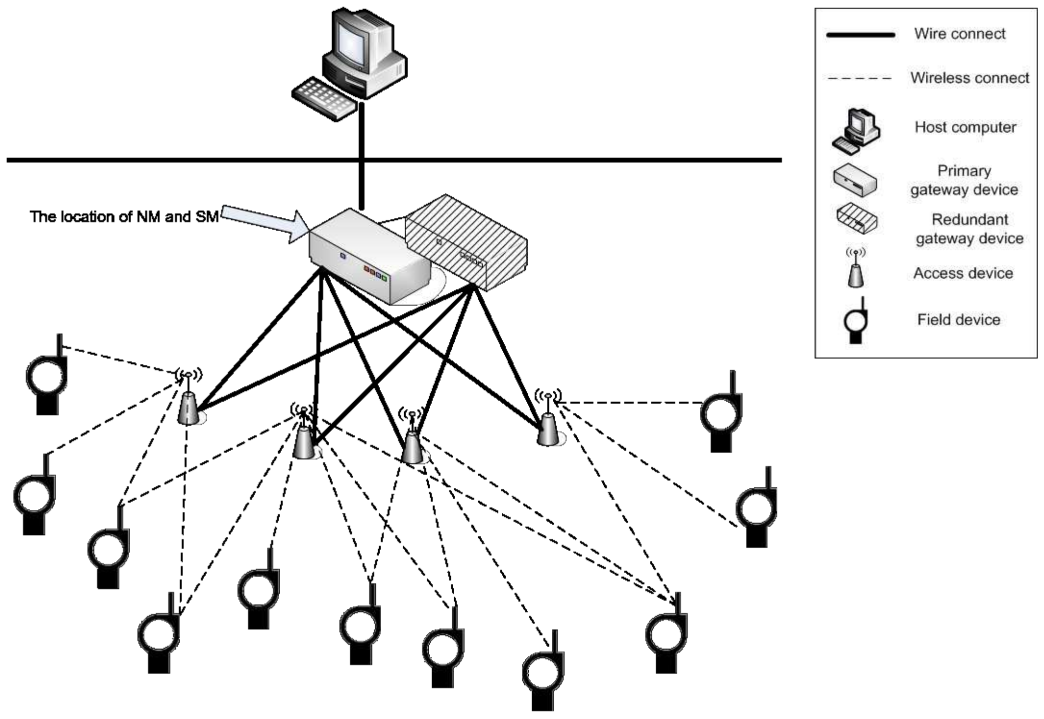

Figure 6 – WIA-FA enhanced star topology

5.3 Protocol architecture

The WIA-FA network protocol follows the ISO/IEC 7498-1 OSI reference model. The WIA-FA network protocol defines the Physical Layer (PHY), Data Link Layer (DLL) and Application Layer (AL). Figure 7 shows the mapping between WIA-FA and OSI basic reference model.

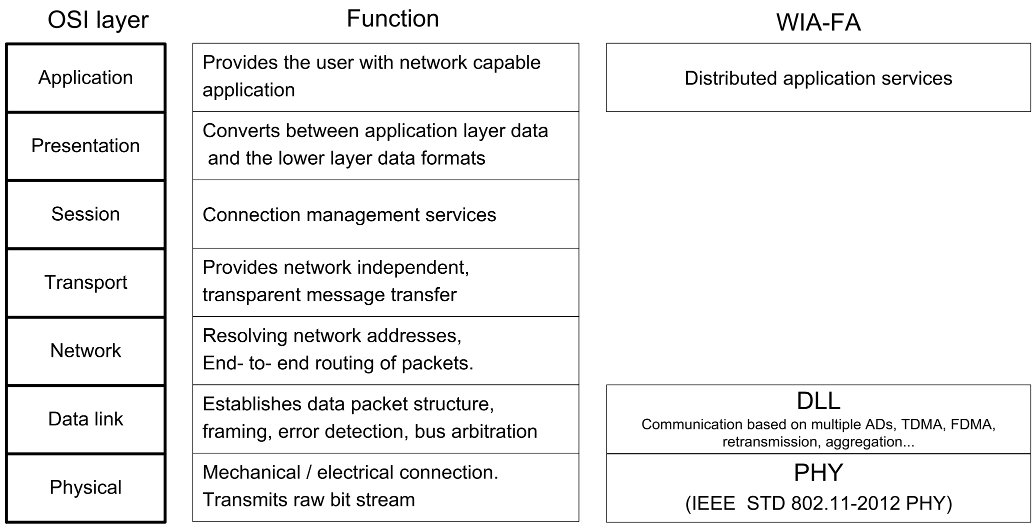
OSI layer
Function
WIA-FA
Application
Provides the user with network capable application
Distributed application services
Presentation
Converts between application layer data and the lower layer data formats
Session
Connection management services
Transport
Provides network independent, transparent message transfer
Network
Resolving network addresses, End- to- end routing of packets.
Data link
Establishes data packet structure, framing, error detection, bus arbitration
DLL
Communication based on multiple ADs, TDMA, FDMA, retransmission, aggregation...
Physical
Mechanical / electrical connection. Transmits raw bit stream
PHY
(IEEE STD 802.11-2012 PHY)

Figure 7 – OSI basic reference model mapped to WIA-FA

Figure 8 shows the protocol architecture of WIA-FA. The protocol architecture of WIA-FA includes the following components:

– Protocol layers: including PHY, DLL, AL; AL is comprised of ASL, UAP and DMAP;

– Entities of protocol layers: including data entities (DLDE and ASLDE) and management entities (DLME and ALSME);

– Protocol layer interfaces: including data entities SAPs (DLDE-SAP and ASLDE-SAP) and management entity SAPs (DLME-SAP and ASLME-SAP).

Device Management Application Process (DMAP) includes the network manager/network management module, security manager/security management module, and MIB. DMAP is a special User Application Process (UAP) and uses ASLDE-SAP together with UAP to exchange messages with ASL.

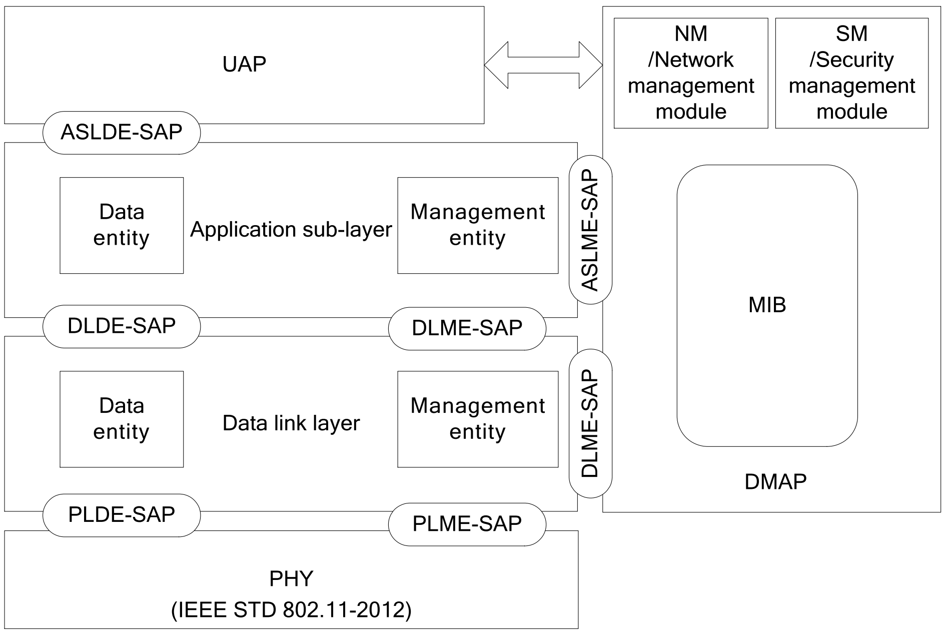

Figure 8 – Protocol architecture of WIA-FA

The data flow over WIA-FA network is shown in Figure 9.

– A field device has PHY, DLL, and AL;

– An access device has PHY and DLL; An access device connects with the gateway device by wires;

– The gateway device only has AL and part of DLL. The application running on the gateway device consists of a component that communicates with the application layer of the WIA-FA network, plus a component that communicates with the application layer of the plant network, plus any components that facilitate translation between the application layer of the WIA-FA network and the application layer of the plant network.

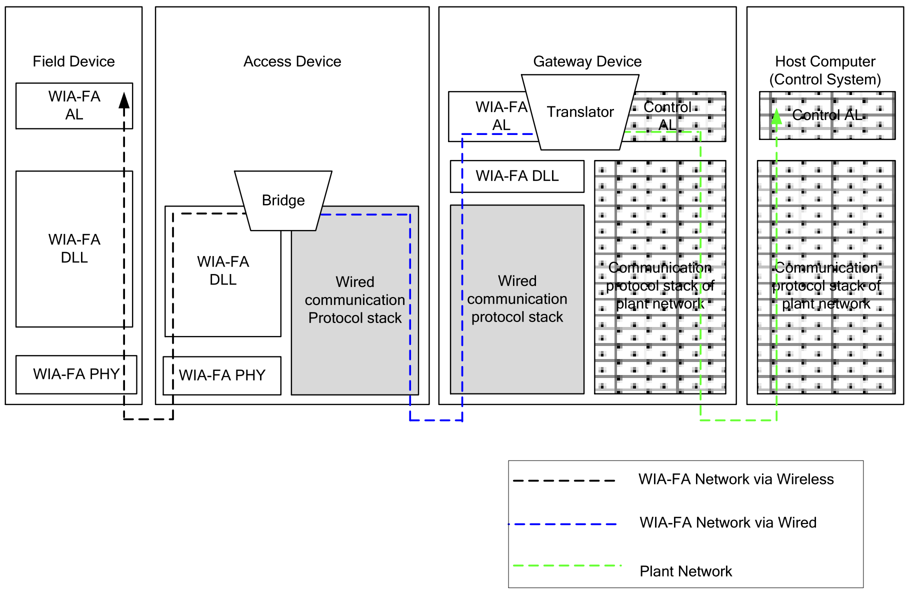
Field Device
WIA-FA AL
WIA-FA DLL
WIA-FA PHY
Access Device
Bridge
WIA-FA DLL
WIA-FA PHY
Wired communication Protocol stack
Gateway Device
WIA-FA AL
Translator
Control AL
WIA-FA DLL
Wired communication protocol stack
Communication protocol stack of plant network
Host Computer (Control System)
Control AL
Communication protocol stack of plant network
WIA-FA Network via Wireless
WIA-FA Network via Wired
Plant Network

Figure 9 – Data flow over WIA-FA network

6 System management

6.1 Overview

The WIA-FA network adopts centralized management framework, as shown in Figure 10. The system management is implemented by the network manager and security manager in the gateway device, and the network management modules and security management modules in field devices and access devices. The network manager and security manager are responsible for managing the access devices and field devices. Network management modules and security management modules are realized in field devices and access devices, which perform management functions together with the gateway device.

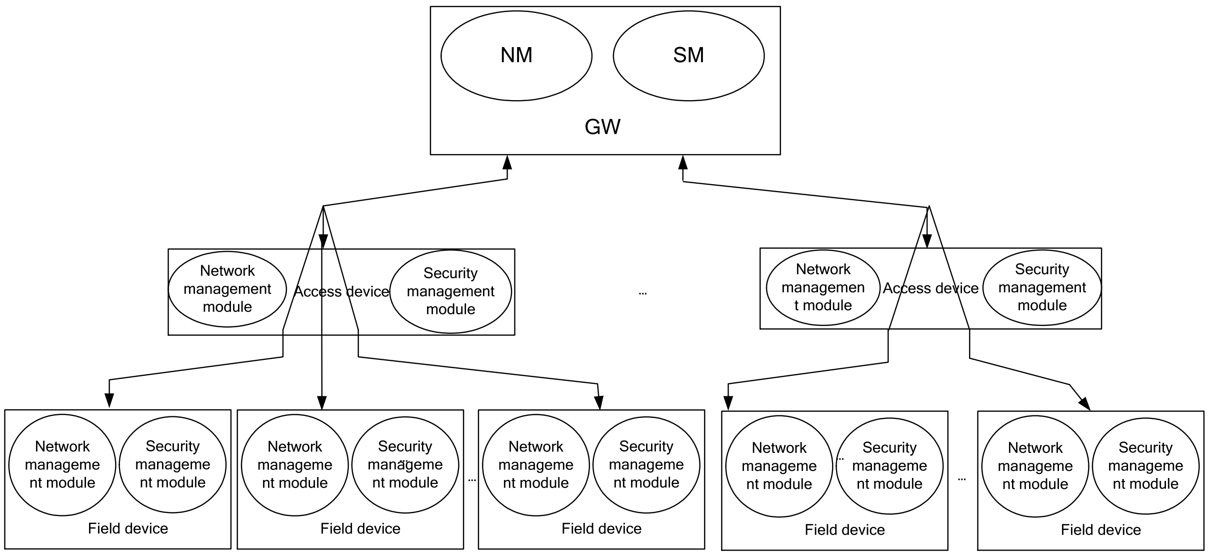
NM
SM
GW
Network management module
Access device
Security management module
...
Network managemen t module
Access device
Security management module
Network manageme nt module
Security manageme nt module
Field device
...

Figure 10 – System management scheme

6.2 Device Management Application Process

The functions of system management in WIA-FA network are implemented by the Device Management Application Process (DMAP) in each device. The DMAP is a particular User Application Process (UAP), responsible for managing devices and providing MIB access services. The position and component of the DMAP in the protocol architecture are shown in Figure 11. The grey part is the DMAP, and the white parts are function modules in DMAP, including:

– Network Manager (NM) of Gateway Device (GW), the network management modules of Access Devices (AD) and Field Device (FD);
– Security Manager (SM) of GW, the security management modules of AD and FD;
– Management Information Base (MIB), saving all the attributes for network management and security management in WIA-FA network.

DMAP is a special User Application Process (UAP) and uses ASLDE-SAP together with UAP to exchange messages with ASL, as shown in Figure 11.

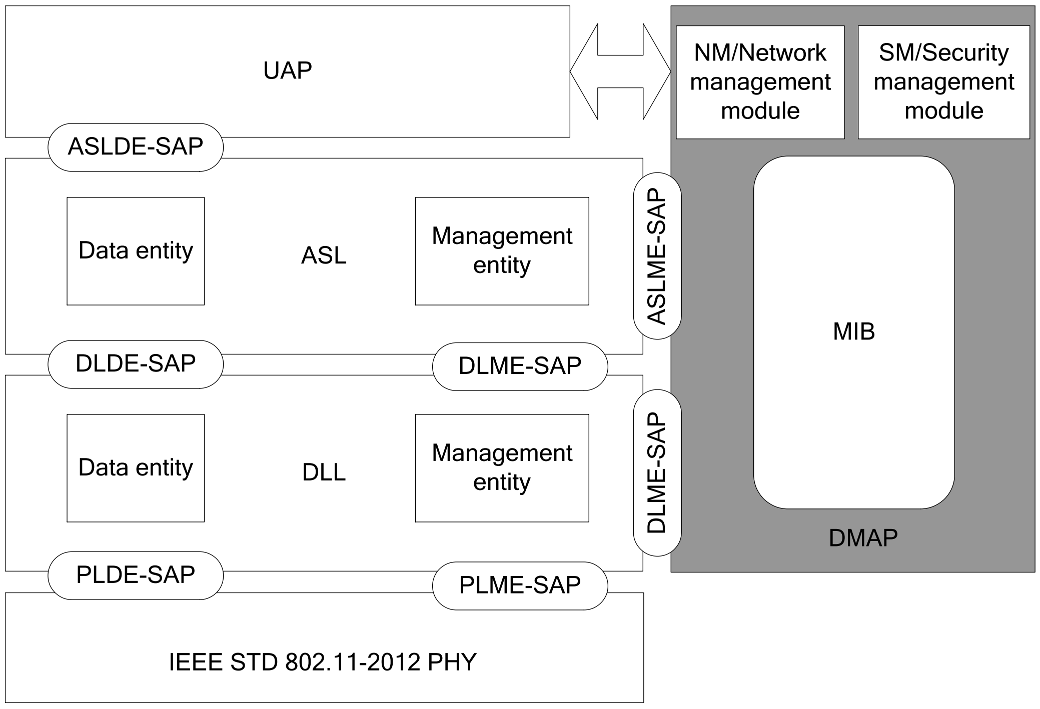

Figure 11 – DMAP of management system

The network management functions implemented in WIA-FA network are shown in Table 8 and
Table 9.

Table 8 – Network management functions

Network management functions	Requirement
Network establishment	Initialization: initializing the NM/network management modules and starting the network
	Time source configuration and system time service; the WIA-FA network should set up one reference time source, which is performed by the gateway device. Devices in the network should only synchronize with gateway device.
	Device join process management: the devices need NetworkID (see Table 14) before joining. The joining devices invoke the join process, after the authentication of SM, the NM returns the join response.
	Network address allocation: Each device in the WIA-FA network has a global unique 64-bit address which is called long address and an 8-bit or 16-bit network address which is called short address. The long address of each device is assigned by vendors according to the 64-bit Extended Unique Identifier (EUI-64). The short address of each network device is assigned by the NM.
	Topology management: forming and maintaining the enhanced star topology as shown in Figure 6.
	Network configuration management: maintaining the communication resources, network address, network attributes, including the information of all network devices distributed by the NM, finishing the configuration of MIB.
Network scheduling and Communication resource allocation	Superframe establishment: establishing the superframe for communication according to the application requirement.
	Communication resource allocation: allocating the communication resources in superframe to the links.
	Activation/deactivation: activating and deactivating the superframe according to application process.
Channel Network diagnosis and performance monitoring	Channel management: monitoring and maintaining the channel list and condition.
	Device health status management: monitoring and maintaining the health status of each device.
	Network performance monitoring
Leaving	Device leave process management: the leave process of field devices and access devices includes passive leaving and abnormal leaving, The passive leaving is invoked by the gateway device, the field devices and access devices leave the network after receiving the leave request, and the gateway device releases the communication resources of field devices; The abnormal leaving is detected and processed by the gateway device.

Table 9 – Security management functions

Security management functions	Requirement
Secure network establishment and configuration	Secure join process: when a new field device invokes the secure join process, SM shall authenticate the field device and return the result to NM; NM returns a join response considering the authentication result.
	Key establishment: after FD secure joining, SM generates and distributes keys used to make secure operation during normal operating process, including KEK, KEDB, and KEDU.
Key updating	Key update: SM updates the keys in use before the ends of their lifetime, including KEK, KEDB, and KEDU.
Security performance monitoring	Security alarm: monitoring the update status and attacked counts of keys.

6.2.1 Network manager

Network Manager (NM) realizes network management function in the gateway device, which manages the information of all devices in the network. One WIA-FA network has only one NM.

NM mainly performs the following functions:

– Allocating the unique 8-bit or 16-bit short address for all devices in the network (see 6.3);
– Constructing and maintaining the enhanced star topology;
– Allocating communication resources for communications of WIA-FA devices;
– Monitoring the performance of the WIA-FA network, including device status, and channel condition, etc.

6.2.2 Security manager

Security Manager (SM) realizes the function of security in the gateway device, and one WIA-FA network has only one SM. SM communicates directly with NM.

SM mainly performs the following functions:

– Authenticating the field devices attempting to join the WIA-FA network;
– Managing keys in the WIA-FA network, including key establishment and update (see 11.4).
– Handling security alarm.

6.2.3 Network management module

Network management modules in the field device and access device maintain the information needed for communication.

The main functions of the network management module are as follows:

– Coordinating with NM to construct and maintain the enhanced star topology;
– Coordinating with NM to allocate communication resources for devices;
– Coordinating with NM to monitor the performance of the WIA-FA network, including device status, and channel condition, etc.

6.2.4 Security management module

Security management modules in the field device and access device maintain the information for security management functions, and realize following security management functions:

– Coordinating with SM to conduct secure joining (field device only);
– Coordinating with SM to manage keys;
– Coordinating with SM to report security alarm.

6.2.5 DMAP state machines

6.2.5.1 DMAP state machine of gateway device

The DMAP state machine of the gateway device is shown in Figure 12, which includes Init and Idle states. The gateway device enters into Active state if it finishes initializations.

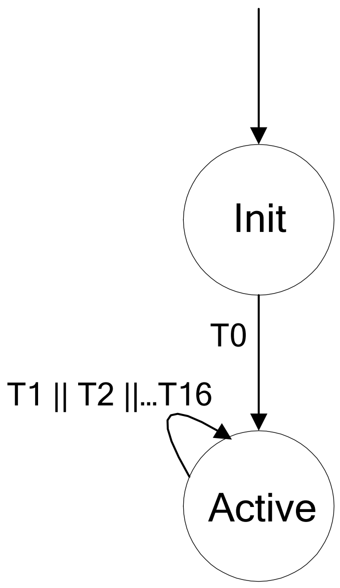

Figure 12 – DMAP state machine of gateway device

The DMAP state transition of gateway device is listed in Table 10.

Table 10 – DMAP state transition of gateway device

#	Current State	Event or condition => action	Next state
T0	Init	IsDMAPInitializationDone() == TRUE => ;	Active
T1|| T2 || … T16	Active	T1|| T2 || … T16 => See Table 11	Active

The DMAP of gateway device maintains a state machine for each field device, as shown in Figure 13. The gateway device can process multiple packets from multiple field devices in parallel. The triggering condition from Init state to Active state in Figure 11 is one of T1 to T16 (see Table 11).

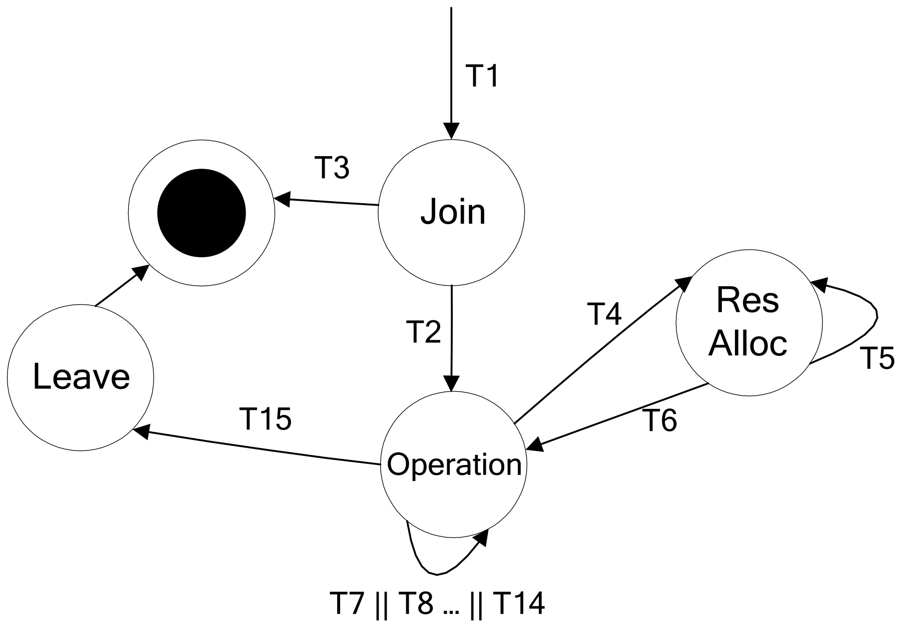

Figure 13 – DMAP state machine of gateway device for each field device

The DMAP state transition of gateway device for each field device is listed in Table 11.

Table 11 – DMAP state transition of gateway device for each field device

#	Current State	Event or condition => action	Next state
T1	Active	PrimitiveType == DLME-JOIN.indication =>	Join
T2	Join	Authentication(PhyAddr, SecMaterial) == SUCCESS && AllocateShortAddr(Addr) == SUCCESS => DLME-JOIN.response(Status:= SUCCESS, ShortAddr);	Operation

T3	Join	Authentication(PhyAddr, SecMaterial) != SUCCESS
		|| AllocateShortAddr(Addr) != SUCCESS
		
		=>
		
		If ((Authentication(PhyAddr, SecMaterial) != SUCCESS)
		
		{
		
		DLME-JOIN.response(Status := AUTH_FAILURE, ShortAddr);
		
		}else if(AllocateShortAddr(Addr) != SUCCESS)
		
		{
		
		DLME-JOIN.response(Status:= NETWORK_SCALE_ERROR, ShortAddr);
		
		}
			End

T4	Operation	IsHostComputerConfigureDone() == TRUE
		
		=>
		
		AllocResult:= ResAllocAgrithm (SuperframeList,LinkList);
		
		If (AllocResult == SUCCESS)
		
		DLME-INFO-SET.request(AttributeOption:= 2, AttributeID:= 131, AttributeMemID:= 12, AttributeValue:= ALLOCATION);
		
		DLME-INFO-SET.request(AttributeOption:= 0, (AttributeID:= 128) || (AttributeID:= 129));
		
		}
			Res Alloc

T5	Res Alloc	PrimitiveType ==DLME-INFO-SET.confirm && (AttributeID == 128 || AttributeID == 129) => DLME-INFO-SET.request(AttributeOption:= 0, (AttributeID:= 128) || (AttributeID:= 129));	Res Alloc
T6	Res Alloc	IsAllResAllocateDone() == TRUE => DLME-INFO-SET.request(AttributeOption:= 2, AttributeID:= 131, AttributeMemID:= 12, AttributeValue:= OPERATION);	Operation

T7	Operation	PrimitiveType == DMAP-MIB-SET.request
		
		=>
		
		Status = WriteToMIB(Handle, AttributeOption, AttributeID, AttributeMemID, FirstStoreIndex, Count, AttributeValue);
		
		DMAP-MIB-SET.confim(Handle, Status);
			Operation

T8	Operation	PrimitiveType == DMAP-MIB-GET.request
		
		=>
		
		Status := ReadFromMIB(Handle, AttributeID, AttributeMemID, FirstStroeIndex, Count, AttributeValue);
		
		DMAP-MIB-GET.confim(Handle, Status, Count, AttributeValue);
			Operation

#	Current State	Event or condition => action	Next state
T9	Operation	IsHostComputerSet MIB() == TRUE => DLME-INFO-SET.request(Handle, DstAddr, AttributeOption, AttributeID, AttributeMemID, FirstStoreIndex, Count, AttributeValue);	Operation
T10	Operation	PrimitiveType ==DLME-INFO-SET.confirm => IndicateSetMIBResult(Handle, Status);	Operation
T11	Operation	IsHostComputerGet MIB() == TRUE => DLME-INFO-GET.request(Handle, DstAddr, AttributeID, AttributeMemID, FirstStoreIndex, Count);	Operation
T12	Operation	PrimitiveType ==DLME-INFO-GET.confirm => IndicateGetMIBResult(Handle, Status);	Operation
T13	Operation	PrimitiveType == DLME-CHANNEL-STATUS.indication => HandleChannelStatusReport(Addr, ChannelConditonInfo);	Operation
T14	Operation	PrimitiveType == DLME-DEVICE-STATUS => HandleDeviceStatusReport(PowerSupplyStatus);	Operation

T15	Operation	IsHostComputerRequestDeviceLeave() == TRUE
		=>
		DLME-LEAVE.request(DeviceAddr);
		ReleaseResources(Addr);
		IndicateHostComputerLeaveResult(Addr);
			Leave

– Join state

In Join state, the DMAP of the gateway device handles join request received from field devices and performs authentication. If a field device is authenticated successfully, the NM in the DMAP assigns short address for the field device. If authentication or short address assignment fails, DMAP invokes DLME-JOIN.response to notify the field device the failure result. After the DMAP state machine enters to End State; if authentication and short address assignment succeeds, the DMAP of the gateway device invokes DLME-JOIN.response to notify the field device that the join process is successful; then, the DMAP state machine enters to Operation state.

– Operation state

In Operation state, the following events shall occur:

a) Host compute remotely sets the field device’s MIB attributes; DMAP invokes DLME-INFO-SET.request to request DLL generating a remote attribute set request frame (see 8.4.16).
b) DLL invokes DLME-INFO-SET.confirm and returns the result of remote attribute set operation to DMAP.
c) Host compute remotely gets the field device’s MIB attributes; DMAP invokes DLME-INFO-GET.request to request DLL generating a remote attribute get request frame (see 8.4.14).
d) DLL invokes DLME-INFO-GET.confirm and returns the related MIB attributes of a field device to DMAP.

e) Host computer invokes DMAP-MIB-SET.request to locally set the gateway's MIB attributes; DMAP sets its MIB attributes and invokes DMAP-MIB-SET.confirm to return the result of local attribute set operation to the host computer.

f) Host computer invokes DMAP-MIB-GET.request to locally get the gateway's MIB attributes; DMAP returns its MIB attributes and invokes DMAP-MIB-GET.confirm to return related MIB attributes to the host computer.

g) DLL invokes DLME-DEVICE-STATUS.indication to DMAP after it receives the device status report from a field device.

h) DLL DLME-CHANNEL-STATUS.indication to DMAP after it receives the channel condition report from a field device.

i) Host computer requests a field device leaving WIA-FA network. DMAP invokes DLME-LEAVE.request to request DLL generating a leave command frame (see 8.4.9).

– Res Alloc State

In Res Alloc state, the DMAP of the gateway device allocates communication resources to a field device; DMAP invokes DLME-INFO-SET.request to remotely writing superframes or links to the field device. If the communication resource allocating process is completed or failure, DMAP remotely sets the field device's DeviceState (see Table 19) attribute to Operation; then, the DMAP state machine enters to Operation State.

– Leave State

In Leave state, DMAP releases all MIB attributes and communication resources that are occupied by a leaving field device; then, the DMAP state machine enters to End state.

6.2.5.2 DMAP state machine of field device

The DMAP state machine of field device is shown in Figure 14.

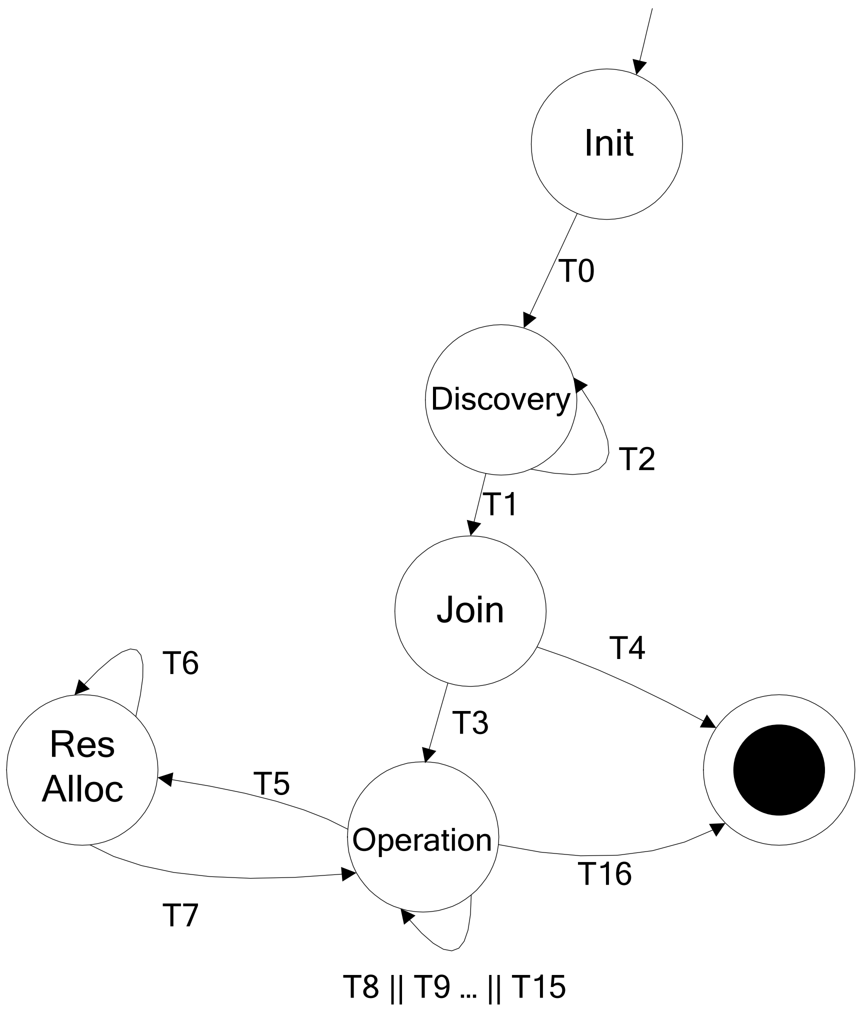

Figure 14 – DMAP state machine of a field device

The DMAP state transition of a field device is shown in Table 12.

Table 12 – DMAP state transition of a field device

#	Current State	Event or condition => action	Next state
T0	Init	IsDMAPInitializationDone() == TRUE => DLME-DISCOVERY.request(ScanChannels, ProbeTime);	Discovery
T1	Discovery	PrimitiveType == DLME-DISCOVERY.confirm && Status == SUCCESS => DLME-JOIN.request(NetworkID, Channel, PhyAddr, SecMaterial);	Join
T2	Discovery	PrimitiveType == DLME-DISCOVERY.confirm &&Status == NO_BEACON => DLME-DISCOVERY.request(ScanChannels, ProbeTime);	Discovery
T3	Join	PrimitiveType == DLME-JOIN.confirm && Status == SUCCESS => DeviceStruct.ShortAddr = ShortAddr;	Operation
T4	Join	PrimitiveType == DLME-JOIN.confirm && Status != SUCCESS =>	End
T5	Operation	PrimitiveType == DLME-INFO-SET.indication && (AttributeID == 131 && AttributeMemID == 12) && AttributeValue == ALLOCATION => Status = WriteToMIB(Handle, AttributeOption, AttributeID, AttributeMemID, FirstStoreIndex, Count, AttributeValue); DLME-INFO-SET.response(Handle, Status);	Res Alloc
T6	Res Alloc	PrimitiveType == DLME-INFO-SET.indication && (AttributeID == 128 || AttributeID == 129) => Status = WriteToMIB(Handle, AttributeOption, AttributeID, AttributeMemID, FirstStoreIndex, Count, AttributeValue); DLME-INFO-SET.response(Handle, Status);	Res Alloc
T7	Res Alloc	PrimitiveType == DLME-INFO-SET.indication && (AttributeID == 131 && AttributeMemID == 12) && AttributeValue == OPERATION => Status = WriteToMIB(Handle, AttributeOption, AttributeID, AttributeMemID, FirstStoreIndex, Count, AttributeValue); DLME-INFO-SET.response(Handle, Status);	Operation

T8	Operation	PrimitiveType == DLME-INFO-SET.indication
		=>
		Status = WriteToMIB(Handle, AttributeOption, AttributeID, AttributeMemID, FirstStoreIndex, Count, AttributeValue);
		DLME-INFO-SET.response(Handle, Status);
			Operation

T9	Operation	PrimitiveType == DLME-INFO-GET.indication
		=>
		Status = ReadFromMIB(Handle, AttributeID, AttributeMemID, FirstStroeIndex, Count, AttributeValue);
		DLME-INFO-GET.response(Handle, Status, Count, FirstStoreIndex, Count, AttributeValue);
			Operation

#	Current State	Event or condition => action	Next state

T10	Operation	PrimitiveType == DMAP-MIB-SET.request
		=>
		Status = WriteToMIB(Handle, AttributeOption, AttributeID, AttributeMemID, FirstStoreIndex, Count, AttributeValue);
		DMAP-MIB-SET.confim(Handle, Status);
			Operation

T11	Operation	PrimitiveType == DMAP-MIB-GET.request
		=>
		Status = ReadFromMIB(Handle, AttributeID, AttributeMemID, FirstStroeIndex, Count, AttributeValue);
		DMAP-MIB-GET.confim(Handle, Status, Count, AttributeValue);
			Operation

T12	Operation	DevStaRptCycle timeout => DLME-DEVICE-STATUS.request(PowerSupplyStatus);	Operation
T13	Operation	PrimitiveType == DLME-DEVICE-STATUS.confirm =>	Operation
T14	Operation	(ChaStaRptCycle timeout) => DLME-CHANNEL-STATUS.request(ChannelConditionInfo);	Operation
T15	Operation	PrimitiveType == DLME-CHANNEL-STATUS.confirm =>	Operation
T16	Operation	PrimitiveType == DLME-LEAVE.indication => ReleaseResources(Addr);	End

– Init state

In Init state, the field device initializes itself and enters to Discovery state.

– Discovery state

In Discover state, the DMAP of a field device invokes DLME-DISCOVERY.request to scan WIA-FA network. DLL invokes DLME-DISCOVERY.confirm to return the result of network scanning. If the result is successful, the DMAP state machine of a field device enters to Join state; otherwise, the DMAP state machine of a field device stays in Discovery state and restarts the network discovery.

– Join state

In Join state, the DMAP of a field device invokes DLME-JOIN.request to try to join the network; DLL invokes DLME-JOIN.confirm to return the joining results. If the result indicates the join process is successful, the state machine enters to Operation state; otherwise, the state machine enters to End state.

– Res Alloc state

In Res Alloc state, DLL invokes DLME-INFO-SET.indication after it receives a remote attribute set request from the gateway device to write superframes or links. The DMAP of the field device writes superframes or links into MIB, and invokes DLME-INFO-SET.response to return the result. If the DeviceState (see Table 19) in MIB is set to Operation by the gateway device, the state machine enters to Operation state.

– Operation state

In Operation state, the following events shall occur:

a) DLL invokes DLME-INFO-SET.indication after it receives a remote attribute set request frame (see 8.4.16), which indicates DMAP to set MIB attributes. The DMAP of a field device sets the MIB attributes properly and invokes DLME-INFO-SET.response to return the result.

b) DLL invokes DLME-INFO-GET.indication after it receives a remote attribute get request frame (see 8.4.14), which indicates DMAP to get MIB attributes. The DMAP of a field device gets the related MIB attributes properly and invokes DLME-INFO-GET.response to return the requested MIB attributes.

c) APP/DLL invokes DMAP-MIB-GET.request to locally get MIB attributes. The DMAP invokes DMAP-MIB-GET.response to return the related MIB attributes.

d) APP/DLL invokes DMAP-MIB-SET.request to locally set the MIB attributes. The DMAP invokes DMAP-MIB-GET.response to return the result of DMAP-MIB-SET.request.

e) If the timer for device status report expires every DevStaRptCycle (see 6.7.1.2.1) time, the DMAP of a field device invokes DLME-DEVICE-STATUS.request to send the device status report to the gateway device.

f) DLL invokes DLME-DEVICE-STATUS.confirm to DMAP, which returns the result of sending device status report.

g) If the timer for the channel condition report expires every ChaStaRptCycle (see 6.7.1.2.1) time, the field device DMAP invokes DLME-CHANNEL-STATUS.request to returns the result of sending channel condition report.

h) DLL invokes DLME-CHANNEL-STATUS.confirm to DMAP, which returns the result of sending channel condition report.

i) DLL invokes DLME-LEAVE.indication after receiving a leave request frame from the gateway device; the field device releases all MIB attributes and communication resources, and enters to End state.

6.2.5.3 Functions used in DMAP state transitions

The functions used in DMAP state transtitions of the gateway device are listed in Table 13.

Table 13 – Functions used in DMAP state transition

Function	Input	Output	Description
Authentication()	PhyAddr	SUCCESS FAILURE	Authenticating a joining field device
AllocateShortAddr()	Addr	SUCCESS FAILURE	Allocating short address for a field device
IsHostComputerConfigureDone()		TRUE FALSE	Judging whether the application configuration of the host computer is finished
ResAllocAgrithm()	SuperframeList LinkList	SUCCESS NO_RESOURCE	Allocating communication resources
IsHostComputerRequestDeviceLeave()		TRUE FALSE	Judging whether the host computer request a field device leave WIA-FA network
IsHostComputerSet MIB()		TRUE FALSE	Judging whether the host computer remotely request set ting the attributes of a field device
IndicateSetMIBResult()	Handle Status		Indicating the results of the remote attribute set operation to the host computer
HandleChannelStatusReport()	Addr ChannelConditonInfo		Processing the channel condition report
HandleDeviceStatusReport()	PowerSupplyStatus		Process the device status report
IsHostComputerGet MIB()		TRUE FALSE	Judging whether the host computer remotely request getting the MIB attributes
IndicateGetMIBResult()	Handle Status		Indicating the results of the remote attribute get operation to the host computer
ReleaseResources()	Addr		Releasing communication resources occupied by a field device
IndicateHostComputerLeaveResult()	Addr		Indicating the leave of a field device to the host computer
WriteToMIB();	Handle AttributeOption AttributeID AttributeMemID FirstStoreIndex Count AttributeValue	SUCCESS INVALIDATTRIBUTE INVALIDATTRIBUTE_ MEMBER INVALID VALUE	Locally writing MIB attributes
ReadFromMIB()	Handle AttributeID AttributeMemID FirstStroeIndex Count AttributeValue	SUCCESS INVALIDATTRIBUTE INVALIDATTRIBUTE_ MEMBER INVALID RANGE	Locally reading MIB attributes

Function	Input	Output	Description
IsDMAPInitializationDone()		TRUE FALSE	Judging whether the DMAP initialization is finished
IsAllResAllocateDone()		TRUE FALSE	Judging whether all communication resources are allocated

6.3 Addressing and address assignment

Each WIA-FA network (field device, access device, and gateway device) has a global unique 64-bit long address and an 8-bit or 16-bit short address (indicated by AddressTypeFlag in Table 14). When the number of field devices in the network is less than 252, 8-bit short address should be adopted; Otherwise, the 16-bit short address should be adopted. The long address, shown in Figure 15, is assigned and set by manufacturers according to the EUI-64. The least significant octet of DeviceShortAddress (see Table 9 in 6.7.1.2.2) is valid if the network uses 8-bit short address.

Gateway device uses AdID to distinguish different access devices.

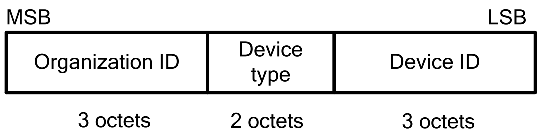

Figure 15 – Long address structure of device

In the WIA-FA network, the 8-bit short address of a WIA-FA device is set as follows:

– The short address of gateway device is 0x01;
– The short address of access device is 0x02;
– The value range of field device short address is 0x03 to 0xFE;
– The broadcast address in WIA – FA network is 0xFF.

In the WIA-FA network, the 16-bit short address of a WIA-FA device is set as follows:

– The short address of gateway device is 0x0001;
– The short address of access device is 0x0002;
– The value range of field device short address is 0x0003 to 0xFFFE;
– The broadcast address in WIA – FA network is 0xFFFF.

The default short address of a device is 0x00 or 0x0000, and it means the device hasn’t been allocated short address.

6.4 Communication resource allocation

6.4.1 General

Communication resources consist of channels (see 3.1.8) and timeslots (see 3.1.31). Allocation of communication resources means allocating channels and timeslots in superframe to access devices and field devices for creating links according to data priorities and resource occupation methods. LinkList attributes include LinkID, LinkType, PeerAddr, RelativeSlotNumber, ChannelIndex, and SuperframeID (see 6.7.1.2.2).

6.4.2 Communication resource allocation

6.4.2.1 Data priorities

According to the functions and requirements of data in industrial fields, the data is set to different priorities. WIA-FA defines the following five data priorities:

– Emergent data (RT0)

  RT0 data has the highest priority. It refers to the data that plays a key role to application behaviours and requires being timely transferred. RT0 data includes: command from the host computer to brake the actuator; emergent alarm of failure/error notification; critical network management services from the host computer, such as start/stop command. The transmission of RT0 data uses R/S communication mode (see 10.5.5.3.3).

– Periodic process data (RT1)

  RT1 data has the second highest priority. It refers to the periodically transmitted process data that has strict real-time requirement. RT1 data mainly includes the physical measurement and control data of a control system. The transmission of RT1 data uses P/S communication mode (see 10.5.5.3.2).

– Aperiodic non-urgent data (RT2)

  RT2 data has the third highest priority. It refers to the aperiodically transmitted data triggered by events, such as the non-urgent alarm data. The transmission of RT2 data uses R/S communication mode (see 10.5.5.3.3).

– Periodic management data (RT3)

  RT3 data has the fourth highest priority. It refers to the periodically transmitted monitoring data of device and network status, which has certain real-time requirements. RT3 data mainly includes the device status, channel state, etc. The transmission of RT3 data uses P/S communication mode (see 10.5.5.3.2).

– Non-real-time data (NRT)

  NRT data has the lowest priority. It refers to data that is generated by network operation and has no strict real-time requirement. In the industrial field, NRT data usually includes parameter configuration and management data. The NRT data transmission should not interfere with transmissions of real-time data. The transmission of NRT data uses C/S communication mode (see 10.5.5.3.1).

The transmission of data with one of above five priorities uses different communication modes and different VCRs. The definition of communication modes, VCR, and the corresponding relationship between VCR and data are shown in 10.5.5.3.

6.4.2.2 Occupation method of communication resources

For transmitting data with different priorities, the occupation methods of communication resources include scheduling, preemption, and competition.

– Scheduling

  Scheduling is used for transmitting RT1 data and RT3 data. Network manager schedules all communication resources for whole WIA-FA network. After field devices and access devices joining the network, the network manager allocates fixed timeslots to periodically transmit and retransmit RT1 and RT3 data.

– Preemption

  Preemption is used to transmit RT0 data. The gateway device and field devices can utilize timeslots original for periodic data to transmit RT0 data, which defers the transmission of the periodic data.

– Competition

  Competition refers to using shared transmitting link to transmit RT2 data and NRT data.

6.5 Joining and leave process of field device

6.5.1 Joining process of a field device

The handheld device provisions a field device through RS232 maintenance port when this field device attempts to join WIA-FA network. The provisioning information includes:

– NetworkID;

– SecLevel;

– KJ (when the security level is not 0);

– KS (when the security level is not 0).

The join process of a provisioned field device is shown in Figure 16. The join process is listed as follows.

– Access devices periodically broadcast beacons;

– A field device with attempt to join the WIA-FA network continually scans available channels to get beacons from access devices and synchronizes with gateway device by using one-way time synchronization method (see 8.1.4);

– The field device chooses a channel on which field device receives beacon frame and utilizes the shared timeslots for sending the join request; the shared timeslot is determined by the "First shared timeslot number" and the “Shared timeslot count” (see 8.4.6) in the beacon frame); the field device competes for transmitting join request utilizing timeslot based backoff method (see 8.1.6.5) on the channel used by the beacon;

– An access device transfers the join request received from a field device to the gateway device;

– The NM residing on the gateway device returns the join response; if the joining is approved, the value of Status in the join response is set to SUCCESS; otherwise, the value of the Status should be set according to error reasons (See Status in 8.3.4.3);

– An access device forwards the join response to the corresponding field device;

– The field device receives the join response from an access device; if the value of Status in the join response is not SUCCESS, the field device should restart or terminate the join process; if the value of Status in the join response is SUCCESS, the join process is completed.

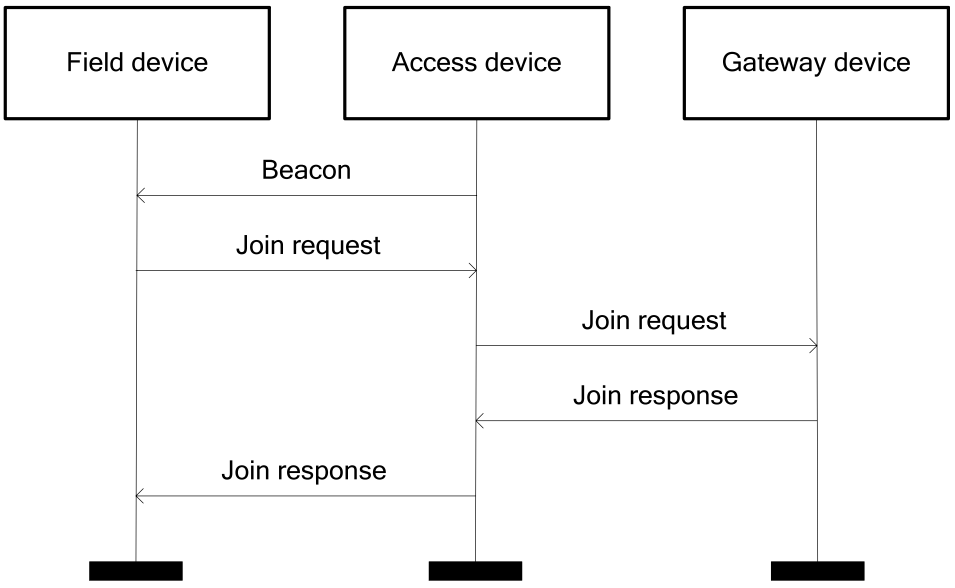

Figure 16 – Joining process of field device

NOTE See Clause 10 for security.

6.5.2 Communication resource allocation to field device

After a new field device joins WIA-FA network, the host computer requests the gateway device to read UAOs of the field device by using the remote attribute get services (see 8.3.7), and to write application configuration information/VCRs (see 10.5.5.3 and 10.5.5.4) for the field device by using the remote attribute configuration services (see 8.3.8).

The NM residing on the gateway allocates communication resources for new joined field device by utilizing the remote attribute configuration services (see 8.3.8). These communication resources are used for communication between field devices and access devices. If the new joined field device influences the superframe structures of access devices (see 8.1.2), the corresponding access device should update its own SuperframeList and LinkList attributes of itself and update SuperframeList and LinkList attributes between it and field devices.

The communication resource allocation process for a field device is illustrated in Figure 17:

– The network manager sends the "remote attribute configuration" request;
– Access devices forward the "remote attribute configuration" request to the field device;
– After receiving the "remote attribute configuration" request, the field device returns the "remote attribute configuration" response;
– Access devices forward the "remote attribute configuration" response to the gateway device.

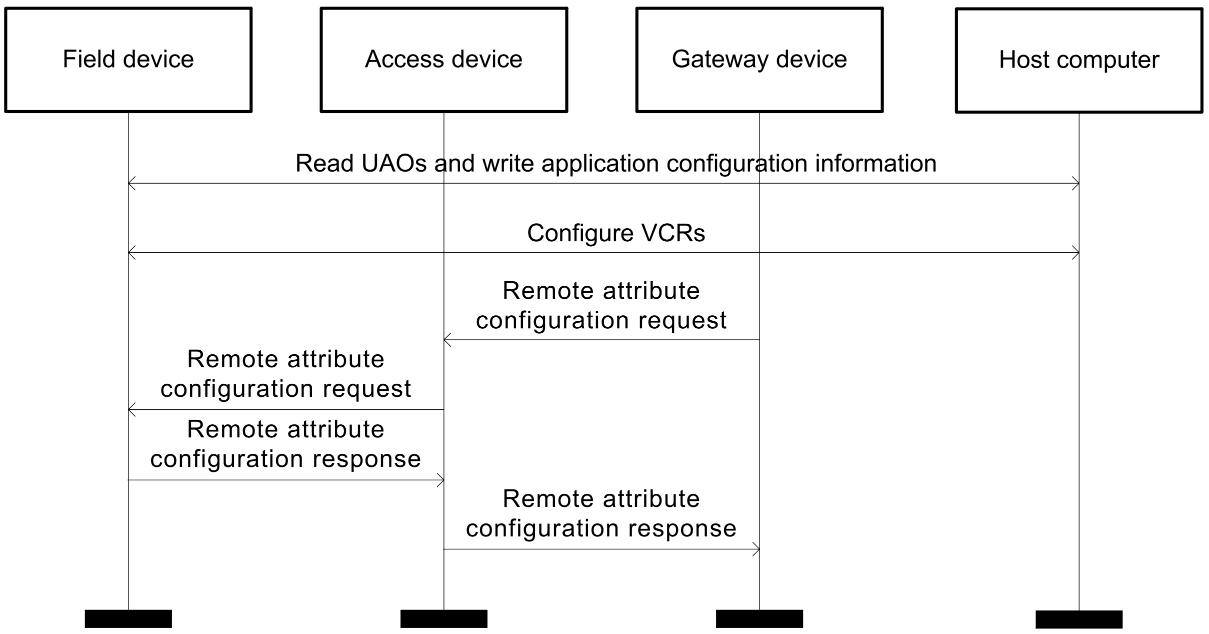
Field device
Access device
Gateway device
Host computer
Read UAOs and write application configuration information
Configure VCRs
Remote attribute configuration request
Remote attribute configuration request
Remote attribute configuration response
Remote attribute configuration response

Figure 17 – Communication resource allocation process for a field device

6.5.3 Leaving process of a field device

The leave process of field devices includes abnormal leaving and passive leaving.

– Abnormal leaving: devices can not communicate with other devices because of failure, invalidation, or energy depletion. If the gateway device does not receive any packet from a field device during LossConnectDuration (see 6.7.1.2.1), it shall judge that the field device has left the WIA-FA network abnormally. The gateway device releases the short address and communication resources of the field device. If the field device does not receive any packet from access devices during LossConnectDuration (see 6.7.1.2.1), it shall judge that its connection with access devices is lost. The field device shall handle it as abnormal leaving and restart itself.

– Passive leaving: the gateway device requests a field device to leave the WIA-FA network. The passive leave process of a field device is shown in Figure 18, which includes the following procedures.

  - The gateway device sends the leave request to a field device (see 8.4.9) through the access device;
  - After receiving the leave request, the field device returns a leave response;
  - After the gateway device receiving the leave response forwarded by access devices from a field device, the network manager on the gateway device handles the leave response, updates the device list, and releases the short address and communication resources of the leaving field device; and the access device updates its MIB accordingly.

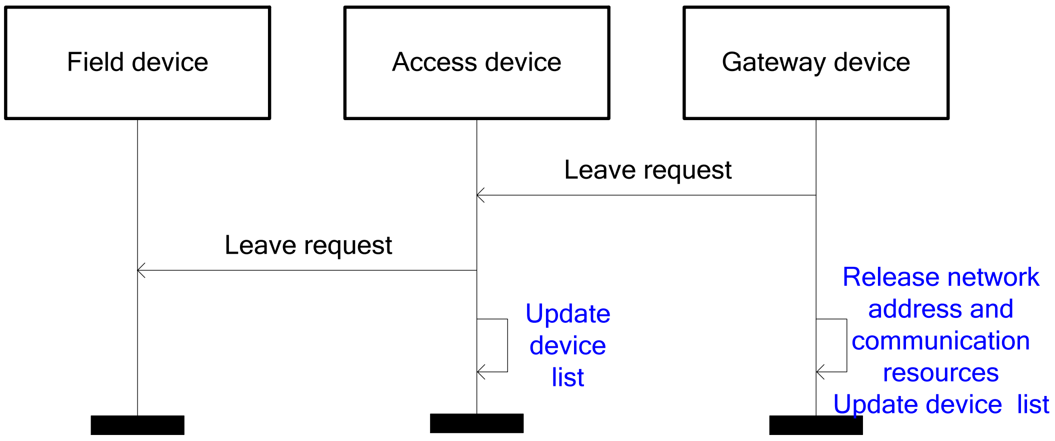

Figure 18 – Passive leave process of a field device

6.6 Network performance monitoring

6.6.1 Device status report

The field device periodically reports its device status to the gateway device (see DeviceList in 6.7.1.2.2 for the detailed device status). After the gateway device receives the device status report of a field device, the NM evaluates and diagnoses the device status. The device status report is used to detect abnormal status of a field device, such as low battery power, etc. The status report process of a field device is shown in Figure 19.

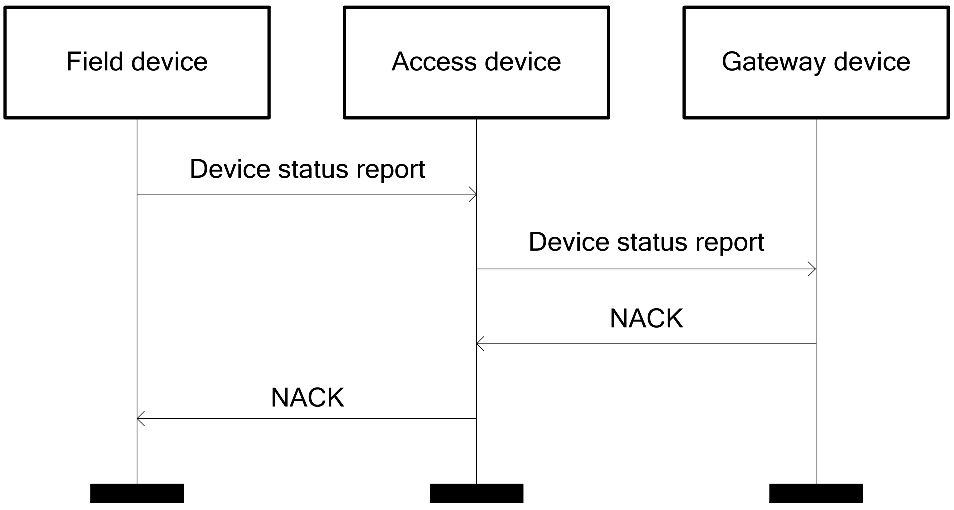

Figure 19 – Device status report process of field device

6.6.2 Channel condition report

The channel condition report is used for a field device to remotely report the channel condition to the NM (see ChannelConditionList in 6.7.1.2.2 for the detailed channel condition). The process of the channel condition report is shown in Figure 20.

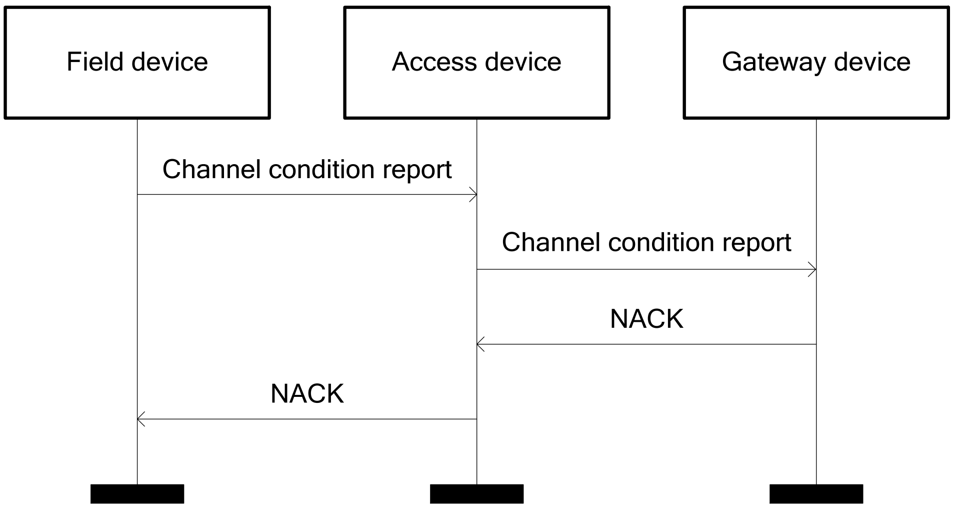

Figure 20 – Channel condition report process of field device

6.7 Management information base and services

6.7.1 Management information base

6.7.1.1 General

Items stored in the MIB are called attributes and are used for monitoring and configuring the WIA-FA network parameters. These attributes can be configured, accessed and updated by the NM.

According to the storage types, the attributes in the MIB are classified into three categories:

– Constant attribute: is unchangeable with time, such as the global unique 64-bit address. The constant attribute is set by manufacturer and should not be modified.

– Static attribute: is changed infrequently during the process of device operation, and can only be modified by NM, such as the period of reporting device status. The static attribute shall hold the previous setting value after the power-down/reboot.

– Dynamic attribute: its value is changed frequently without any external command. The dynamic attribute shall recover to the default value set by manufacturer after the power-down/reboot.

According to the attribute data types, the attributes in the MIB are divided into unstructured attributes and structured attributes.

There are two types of access control to attributes in the MIB:

R (Read), which means the value of the attribute can be read by other devices in WIA-FA network; and

W (Write), which means that the value of attribute can be modified by other devices in WIA-FA network.

According to the implementation requirements, the attributes in the MIB are divided into mandatory attributes and optional attributes.

Attributes in MIB are identified by AttributeID.

6.7.1.2 MIB attributes

6.7.1.2.1 Unstructured attributes

Unstructured attributes are listed in Table 14. Unstructured attributes are unique in whole WIA-FA network.

Table 14 – Unstructured attributes

Attribute ID	Attribute name	Data type	Valid range	Access type	Storage type	Default value	Device type	Description
0	PriGwFailureTime	TimeData	0 to (2^{64}−1)	R/W	S	10	Gateway device	The maximal duration without heartbeat signal (in μs)
1	AddressTypeFlag	Unsigned8	0 to 255	R/W	S	0	All devices	0 = 8-bit short address; 1 = 16-bit short address
2	MaxPayloadLength	Unsigned16	0 to (2^{16}− 1)	R/W	S	1000	All devices	Maximum length of the DLL payload (in octet)
3	NACKCount	Unsigned8	0 to 255	R/W	S	1	All devices	Number of broadcast NACKs
4	NetworkID	Unsigned8	0 to 255	R/W	S	0	All devices	Network identifier, used for identifying multiple coexisting networks
5	BitMap	Bit Field	0 to (2^{24}− 1)	R/W	S	0	All devices	Channel bitmap, indicating used modulation mechanism and channel states: bit 0 to 3: modulation mechanism; bit 4 to 17: channel state, 0 stands for unavailable channel, 1 stands for available channel; others are reserved. See 7.3.3 for details of BitMap coding.
6	DevStaRptCycle	Unsigned16	0 to 65 535	R/W	S	10	Gateway devices and field devices	Cycle of device status report (in length of default superframe)
7	ChaStaRptCycle	Unsigned16	0 to 65 535	R/W	S	10	Gateway devices and field devices	Cycle of channel status report (in length of default superframe)
8	LossConnectDuration	Unsigned24	0 to (2^{24}− 1)	R/W	S	30	Gateway device and field devices	If a WIA-FA device does not receive any packet from a neighbour device during LossConnectDuration, it shall judge that the neighbour device has abnormally left the WIA-FA network (in length of default superframe)
9	KeyupdataDuration	Unsigned8	0 to 255	R/W	S	24	All devices	Cycle of key update (in hour)
10	TimeSlotDuration	Unsigned16	0 to 65 535	R/W	S	200	All devices	Timeslot length (in μs)

Attribute ID	Attribute name	Data type	Valid range	Access type	Storage type	Default value	Device type	Description
11	TwoWayTimeSyn	Unsigned8	0 to 255	R/W	S	0	All devices	Indicate the utilization of two-way time synchronization method: 0 = one-way time synchronization; 1 = two-way time synchronization.
12	TwoWayOverTime	Unsigned8	0 to 255	R/W	S	1	All devices	The dead time of the two-way time synchronization ((in length of default superframe)
13	ADTeamNum	Unsigned8	0 to 255	R/W	S	1	Gateway device	Count of the AD teams
14	TargetLossRate	Single Float	0 to 255	R/W	S	0	Gateway device	Expected packet loss rate in WIA-FA network, which is between 0 and 1
15	LossRate	Single Float	0 to 255	R/W	S	0	Gateway device	Current packet loss rate of factory environment, which is between 0 and 1.
16	MaxRetry	Unsigned8	0 to 255	R/W	S	3	All devices	Maximum number of retransmissions

17	SecLevel	Unsigned8	0 to 255	R/W	S	0	All devices	0 = None;
								1 = Authentication;
								2 = Authentication&MIC-32;
								3 = Authentication&MIC-64;
								4 = Authentication&MIC-128;
								5 = Authentication&ENC;
								6 = Authentication&ENC&MIC-32;
								7 = Authentication&ENC&MIC-64;
								8 = Authentication&ENC&MIC-128;
								others are reserved

18	AttackStatisDur	Unsigned16	0 to 65 535	R/W	S	60	All devices	The period of attack statistics (in min)
19	MaxKeyAttackedNum	Unsigned8	0 to 255	R/W	S	5	All devices	Maximum count of key attacked
20	AlarmRptDur	Unsigned8	0 to 255	R/W	S	1	All devices	Interval of repeating alarm (in length of default superframe)

6.7.1.2.2 Structured attributes

Structured attributes are listed in Table 15. Access device and field device should only maintain structured attributes related to them. Gateway device should maintain structured attributes for each field device, and stores structured attributes of each device indexed by short address or AdID.

Table 15 – Structured attributes

Attribute ID	Attribute name	Data type	Access type	Storage type	Device type	Description
128	SuperframeList	Superframe_Struct structure, see Table 16	R/W	D	All devices	Describing the superframe information
129	LinkList	Link_Struct structure, see Table 17	R/W	D	All devices	Describing the link information
130	ChannelConditionList	ChanCon_Struct structure, see Table 18	R/W	D	Gateway device and field devices	Recording the statistic information of channel condition
131	DeviceList	Device_Struct structure, see Table 19	R/W	D	All devices	Describing the device-related attributes of WIA-PA devices
132	KeyList	Key_Struct structure, see Table 20	R/W	D	All devices	Describing the information of keys.
133	VCRList	VcrEP_Struct structure, see Table 21	R/W	D	Gateway device and field devices	Recording the VCR information
134	SupUAOList	UAOClassDesc _Struct structure, see Table 22	R	S	Gateway device and field devices	Describing the UAP information of each field device.
135	CfgUAOList	UAPInstDesc _Struct structure, see Table 23	R/W	S	Gateway device and field devices	Describing the UAP information of each configured field device.

Table 16 – Superframe_Struct structure

MemberID	Member name	Data type	Valid range	Description
0	SuperframeID	Unsigned8	0 to 255	Unique identifier of the superframe, supplied by the NM
1	NumberSlots	Unsigned16	0 to 65 535	Superframe size (counts of timeslots)
2	ActiveFlag	Unsigned8	0 to 255	Superframe active flag: 0 = Inactive; 1 = Active.
3	ActiveSlot	Unsigned48	0 to (2^{48}–1)	Absolute timeslot number (ASN) when a superframe begins active, which is calculated as: $\lfloor$TimeValue/TimeSlotDuration$\rfloor$.

Table 17 – Link_Struct structure

MemberID	Member name	Data type	Valid range	Description
0	LinkID	Unsigned16	0 to 65 535	Unique identifier of the link

1	LinkType	Unsigned8	0 to 255	Bit 0 represents the link type:
				0 = Unicast;
				1 = Broadcast;
				Bit 1 and bit 2 represent the character of a link:
				00 = Transmitting;
				01 = Transmit-shared;
				10 = Retransmitting;
				11 = Receiving;
				Bit 3 to bit 5represents the type of a timeslot:
				000 = Beacon;
				001 = NACK;
				010 = GACK;
				011 = Management timeslot;
				100 = Data timeslot;
				101 = Management/Data timeslot;
				others are reserved.
				Bit 6 to bit 7 are reserved.

2	ActiveSlot	Unsigned48	0 to ($2^{48}-1$)	Absolute timeslot number (ASN) when a link begins active, which is calculated as: $\lfloor$TimeValue/TimeSlotDuration$\rfloor$.
3	PeerAddr	Unsigned16	0 to 65 535	The short address of peer device
4	RelativeSlotNumber	Unsigned16	0 to 65 535	Relative timeslot number
5	ChannelIndex	Unsigned8	0 to 255	The sequence numbers of current used channel. 0 to 13 is used in this PAS; others are reserved.
6	SuperframeID	Unsigned8	0 to 255	Reference to an superframe in the superframe table

Table 18 – ChanCon_Struct structure

MemberID	Member name	Data type	Valid range	Description
0	ChannelID	Unsigned8	0 to 255	The sequence number of channel. 0 to 13 is used in this PAS; others are reserved.
1	LinkQuality	Unsigned8	0 to 255	Link Quality Indication (LQI) value of a channel
2	PacketLossRate	Single Float	0 to 255	Packet loss rate of a channel, the value of which is between 0 and 1
3	RetryNum	Unsigned16	0 to 65 535	The count of retransmission of every channel

Table 19 – Device_Struct

Member ID	Member name	Data type	Valid range	Access type	Storage type	Default value	Description
0	Version	Unsigned16	0 to 65535	R	C	2013	Device version, owing to different devices

1	LongAddress	Unsigned64	0 to (2^{64}–1)	R	C		EUI-64 (see 6.3), which defines the octet 4 and octet 5 as follows:
							0 = Gateway device;
							1 = Access device;
							2 = Field device;
							3 = Handheld device.

2	AGGSupportFlag	Unsigned8	0 to 255	R/W	S	0	Aggregation and disaggregation support flag (whether a routing device supports aggregation mechanism): 0 = Not support; 1 = Support.
3	AGGEnableFlag	Unsigned8	0, 1	R/W	D	0	Aggregation and disaggregation enable flag: 0 = Disenable; 1 = Enable.
3	NumOfSupUAO	Unsigned16	0 to 65 535	R	S	1	Number of UAOs supported by a field device
4	NumOfCfgUAO	Unsigned16	0 to 65 535	R/W	S	1	Number of configured UAOs in a field device
5	TxDelay	Unsigned16	0 to 65535	R/W	D	1200	Transmission delay time of a frame (in μs)
6	ProbeTime	Unsigned8	0 to 255	R/W	S	2	Time for scanning a channel (in length of default superframe)
7	TimeValue	TimeData	0 to ($2^{64}-1$)	R/W	D	0	Absolutely time (in μs), counted from 0
8	RedundantDevFlag	Unsigned8	0 to 255	R/W	S	0	Flag that indicates whether this device is a redundant device: 0 = Irredundant device; 1 = Redundant device.
9	AdID	Unsigned8	0 to 255	R/W	S	0	Identifier of an access device, invalid for a field device
10	DeviceShortAddress	Unsigned16	0 to 65 535	R/W	S	0	Short device of a device (see 6.3)
11	PowerSupplyStatus	Unsigned8	0 to 255	R/W	S	10	Types of power supply and energy levels: 0 = Fixed power supply; 1to10 = energy level of battery power supply(from low to high)

12	DeviceState	Unsigned8	0 to 3	R/W	D	0	Status of device:
							0 = Not joined;
							1 = Joining;
							2 = Security authenticating;
							3 = Application configuring;
							4 = Resource allocating;
							5 = Operation;
							others are reserved.

Table 20 – Key_Struct structure

MemberID	Member name	Data type	Valid range	Description
0	KeyID	Unsigned16	0 to 65 535	Key identifier
1	PeerAddr	Unsigned16	0 to 65 535	Pair address, namely the short address of neighbour device

2	KeyType	Unsigned8	0 to 255	Key type:
				0 = KJ;
				1 = KS;
				2 = KEK;
				3 = KEDU;
				4 = KEDB;
				Others are reserved.

3	KeyDataValue	KeyData		Key value
4	KeyActiveSlot	Unsigned48	0 to $(2^{48}-1)$	Absolute timeslot number (ASN) when key begins active.(In milliseconds)
5	KeyAttackCnt	Unsigned16	0 to 65 535	The total number of key attacks
6	AlarmFlag	Unsigned8	Bitmap	security alarm event flag related to the key are detected. If a security event related to the key is detected, the corresponding bit is set to 1: Bit 0: Key attacked alarm; Bit 1: Key update timeout alarm.

7	KeyState	Unsigned8	0 to 255	The using state of a key:
				0 = BACKUP;
				1 = USING;
				2 = EXPIRED;
				3 = INVALID;
				Others are reserved.

Table 21 – VcrEP_Struct definition

MemberID	Member name	Data type	Valid range	Description
0	VCR_ID	Unsigned16	0 to 65 535	Unique identifier of the VCR on the device. VCR_ID = 0 is used for default C/S VCR, others are configured by host computer.

1	VcrEP_Type	Unsigned8	0 to 255	Type of the VCR endpoint:
				0 = CLIENT;
				1 = SERVER;
				2 = PUBLISHER;
				3 = SUBSCRIBER;
				4 = REPORT SOURCEt;
				5 = REPORT SINK;
				others are reserved.

2	UAP_ID	Unsigned8	0 to 255	Unique identifier of the UAP on the device. UAP_ID = 0 is used for DMAP, others are configured by host computer.
3	PeerAddr	Unsigned16	0 to 65 535	Short address of the peer field device or gateway device.
4	VCRActiveTime	TimeData	0 to (2^{64}—1)	Only valid for P/S VCR, indicating the absolute time when the VCR endpoint should be activated. The DataUpdateRate of the UAP shall start at this time. The default value is 0, which means the VCR endpoint shall be activated immediately. For C/S and R/S VCRs, the value should be set 0.
5	DataUpdateRate	Unsigned32	0 to (2^{32}—1)	Only valid for P/S VCR, indicating the process data publishment cycle of UAP (in ms). For C/S and R/S VCRs, the value should be set 0.
6	Deadline	Unsigned8	0 to 255	Only valid for P/S VCR, indicating the maximum amount of DataUpdateRate for the VCR endpoint not receiving new data from the last time The max. time = DataUpdateCycle × Deadline. If the device has not received new data within Deadline time interval, it shall produce the "PROCESS DATA NOT UPDATED" alarm event.
7	WatchdogTime	Unsigned32	0 to (2^{32}—1)	Only valid for C/S VCR endpoint, indicating the maximum time that the VCR endpoint shall wait for the service response (in ms). The default value is 100 ms. If the VCR endpoint has not receive the service response within WatchdogTime interval, it shall returns a negative response with "Service time expired"

Table 22 – UAOClassDesc_Struct definition

MemberID	Member name	Data type	Valid range	Description
0	ClassID	Unsigned8	0 to 255	Unique identifier of the UAO class on the field device, indicating the class template for instantiating UAOs.

1	UAOTypte	Unsigned8	0 to 255	Type of the UAO class, the values are as follows:
				0 = AI;
				1 = AO;
				2 = DI;
				3 = DO;
				others are reserved.

2	MaxInputDataLen	Unsigned8	0 to 255	Maximum input data size supported by the UAO class
3	MaxOutputDataLen	Unsigned8	0 to 255	Maximum output data size supported by the UAO class
4	MinDataUpdateRate	Unsigned32	0 to $(2^{32}-1)$	Minimum process data publishment cycle (in ms) supported by the UAO class
5	SuppInputType	ProDataDesc_Struct, see Table 23		Input data description of the UAO, indicating all data types of the input data supported by the UAO class. That is, the bits representing the input data types supported by the UAO class shall be set 1. If the UAO class has no input data, each bit shall be set 0.
6	SuppOutputType	ProDataDesc_Struct, see Table 23		Output data description of the UAO, indicating all data types of the output data supported by the UAO class. That is, the bits representing the output data types supported by the UAO class shall be set 1. If the UAO class has no output data, each bit shall be set 0.

Table 23 – ProDataDesc_Struct definition

MemberID	Member name	Data type	Data Length (in octet)	Valid range	Description
0	ParamDesc	Bit Field	2	0 to 65 535	Indicating the process data type supported by UAO. Each bit represents a data type and the coding is as follows: Bit0 = Unsigned8, fixed length 1 octet Bit1 = Unsigned16, fixed length 2 octets Bit2 = Unsigned32, fixed length 4 octets Bit3 = DigitalData8, fixed length 2 octets Bit4 = DigitalData16, fixed length 3 octets Bit5 = DigitalData32, fixed length 5 octets Bit6 = Single Float, fixed length 4 octets Bit7 = Double Float, fixed length 8 octets Bit8 = SingleAnalogData, fixed length 5 octets Bit9 = DoubleAnalogData, fixed length 9 octets Bit10 = Octetstring The value of Bit0 to Bit10 is as follows: 0 = not supported 1 = supported

Table 24 – UAOInstDesc_Struct definition

MemberID	Member name	Data type	Valid range	Description
0	UAO_ID	Unsigned8	0 to 255	Unique identifier of the UAO on the field. UAO_ID = 0 is used for MIB
1	Class_ID	Unsigned8	0 to 255	Class identifier of the UAO class, indicating the UAO is an instance of the UAO class with class_ID in the SuppUAOList
2	UAP_ID	Unsigned8	0 to 255	Identifier of the UAP that the UAO belonging to on the device. If a UAP is allocated with more than one UAO, these UAOs shall have the same UAP_ID
3	AckFlag	Unsigned16		This value shall be set to the AckFlag of the UAO EventData. See Table 80 for the coding of each bit.
4	NumInputData	Unsigned8	0 to 255	Account of UAO input data
5	NumOutputData	Unsigned8	0 to 255	Account of UAO output data
6	CfgInputDataList	ProDataDesc_Struct, see Table 23		Data type description list of the UAO input data. Each member of the list indicates one process data type and the member order specifies the process data order to be transferred periodically. If Bit 10 is set 1, Bit 0 to Bit 5 indicate the octet length of the Octetstring data as follows: Bit 0 to Bit 5 = 0, 1 octet; Bit 0 to Bit 5 = 1, 2 octets; … Bit 0 to Bit 5 = 31, 32 octets; others are reserved.
I7	CfgOutputDataList	ProDataDesc_Struct, see Table 23		Data type description list of the UAO output data. Each member of the list indicates one process data type and the member order specifies the process data order to be transferred periodically. If Bit 10 is set 1, Bit 0 to Bit 5 indicate the octet length of the Octetstring data as follows: Bit 0 to Bit 5 = 0, 1 octet; Bit 0 to Bit 5 = 1, 2 octets; … Bit 0 to Bit 5 = 31, 32 octets; others are reserved.

6.7.2 MIB services

6.7.2.1 General

The attributes in the MIB can be read and written locally through the DMAP attribute get and DMAP attribute set services provided by local DMAP.

6.7.2.2 DMAP attribute get service

DMAP-MIB-GET.request is used by all layers to request attributes in the MIB.

The semantics of DMAP-MIB-GET.request are as follows:

```
DMAP-MIB-GET.request(
                    Handle,
                    ShortAddr,
                    AttributeID,
                    MemberID,
                    FirstStoreIndex,
                    Count
                     )
```

Table 25 specifies the parameters for DMAP-MIB-GET.request.

Table 25 – DMAP-MIB-GET.request parameters

Parameter	Data type	Valid range	Description
Handle	Unsigned8	0 to 255	Assigned handle when invoking the DMAP-MIB-GET.request
ShortAddr	Unsigned16	0 to 65 535	The 8 or 16-bit short address of a field device, or the AdID (see 6.7.1.2.1) of an access device. ShortAddr is valid only for GW to read its local MIB attributes of a field device or an access device.
AttributeID	Unsigned8	0 to 255	Attribute ID in the MIB
MemberID	Unsigned8	0 to 255	The identifier of attribute member. The value 255 means that all attributes should be read. MemberID is invalid for unstructured attributes.
FirstStoreIndex	Unsigned16	0 to 65 535	The first storage index of multiple attribute values. FirstStoreIndex is invalid for unstructured attributes.
Count	Unsigned16	0 to 65 535	Number of attribute values or attributes member values, which is only used to get the structured MIB attributes; Getting all attribute values from FirstStoreIndex if Count = 0

DMAP-MIB-GET.confirm is used to return the result of DMAP-MIB-GET.request.

The semantics of DMAP-MIB-GET.confirm are as follows:

```
DMAP-MIB-GET.confirm(
                    Handle,
                    Status,
                    Count,
                    AttributeValue
                    )
```

Table 26 specifies the parameters for DMAP-MIB-GET.confirm.

Table 26 – DMAP-MIB-GET.confirm parameters

Parameter	Data type	Valid range	Description
Handle	Unsigned8	0 to 255	Assigned handle when invoking the DMAP-MIB-GET.request

Status	Unsigned8	0 to 255	Attribute getting results:
			0 = SUCCESS;
			1 = INVALIDATTRIBUTE;
			2 = INVALIDATTRIBUTEMEMBER;
			3 = INVALID RANGE;
			Others are reserved.

Count	Unsigned16	0 to 65 535	Number of attribute values or attributes member values, which is only used to get the structured MIB attributes; Getting all attribute values from FirstStoreIndex if Count = 0. Count is valid only if Status = SUCCESS
AttributeValue	Octetstring		Returned attribute values or attributes member values. AttributeValue is valid only if Status = SUCCESS.

If the operation of getting attributes is successful, the Status should be SUCCESS and the AttributeValue is valid; if the MIB does not have the needed attributes, the Status should be INVALID ATTRIBUTE; if the MIB does not have the needed attribute members, the Status should be INVALID ATTRIBUTE MEMBER; if the MIB does not have the needed records indexed with FirstStoreIndex and Count, the Status should returns INVALID RANGE.

6.7.2.3 DMAP attribute set service

DMAP-MIB-SET.request is used by protocol layers to write attributes to the MIB.

The semantics of DMAP-MIB-SET.request are as follows:

```
DMAP-MIB-SET.request(
                    Handle,
                    ShortAddr,
                    AttributeID,
                    MemberID, FirstStoreIndex,
                    Count,
                    AttributeValue
                    )
```

Table 27 specifies the parameters for DMAP-MIB-SET.request.

Table 27 – DMAP-MIB-SET.request parameters

Parameter	Data type	Valid range	Description
Handle	Unsigned8	0 to 255	Assigned handle when invoking the DMAP-MIB-SET.request
ShortAddr	Unsigned16	0 to 65 535	The 8 or 16-bit short address of a field device, or the AdID (see 6.7.1.2.1) of an access device. ShortAddr is valid only for GW to write its local MIB attributes of a field device or an access device.
AttributeID	Unsigned8	0 to 255	Attribute ID in the MIB
MemberID	Unsigned8	0 to 255	The identifier of attribute member. The value 255 means that all attributes should be written. MemberID is invalid for unstructured attributes.
FirstStoreIndex	Unsigned16	0 to 65 535	The first storage index of multiple attribute values. FirstStoreIndex is invalid for unstructured attributes.
Count	Unsigned8	0 to 255	Number of attribute values or attributes member values, which is used to set the structured MIB attributes; Setting all attribute values from FirstStoreIndex if Count = 0
AttributeValue	Octetstring		Written attribute values or attributes member values.

DMAP-MIB-SET.confirm is used to return the result of DMAP-MIB-SET.request.

The semantics of DMAP-MIB-SET.confirm are as follows:

```
DMAP-MIB-SET.confirm(
                         Handle,
                         Status
                          )
```

Table 28 specifies the parameters for DMAP-MIB-SET.confirm.

Table 28 – DMAP-MIB-SET.confirm parameters

Parameter	Data type	Valid range	Description
Handle	Unsigned8	0 to 255	Assigned handle when invoking the DMAP-MIB-SET.request

Status	Unsigned8	0 to 255	Attribute setting results:
			0 = SUCCESS；
			1 = INVALID ATTRIBUTE;
			2 = INVALID ATTRIBUTE MEMBER;
			3 = INVALID VALUE;
			Others are reserved.

If the operation of setting attributes is successful, the Status should be SUCCESS; if the MIB does not have the needed attributes, the Status should be INVALID ATTRIBUTE; if the MIB does not have the needed attribute members, the Status should be INVALID ATTRIBUTE MEMBER; if AttributeValue is out of valid range, the Status should be INVALID VALUE.

7 Physical layer

7.1 General

The physical layer of WIA-FA is based on the IEEE STD 802.11-2012 PHY. WIA-FA supports different modulation modes (FHSS, DSSS, OFDM, etc.) in IEEE STD 802.11-2012.

7.2 General requirements based on IEEE STD 802.11-2012

Table 29 specifies the IEEE STD 802.11-2012 PHY selection for a WIA-FA device.

Table 29 – PHY protocol selection

Clause	Header	Presence	Constraints
7	PHY service specification		
7.1	Scope	YES	
7.2	PHY functions	YES	
7.3			
7.3.1	Scope and field of application	YES	
7.3.2	Overview of the service	YES	
7.3.3	Overview of interactions	YES	
7.3.4			
7.3.4.1	General	YES	
7.3.4.2	PHY-SAP peer-to-peer service primitives	YES	
7.3.4.3	PHY-SAP sublayer-to-sublayer service primitives	YES	
7.3.4.4	PHY-SAP service primitives parameters	YES	
7.3.4.5	Vector descriptions	YES	
7.3.5			
7.3.5.1	Introduction	YES	
7.3.5.2	PHY-DATA.request	YES	
7.3.5.3	PHY-DATA.indication	YES	
7.3.5.4	PHY-DATA.confirm	YES	
7.3.5.5	PHY-TXSTART.request	YES	
7.3.5.6	PHY-TXSTART.confirm	YES	
7.3.5.7	PHY-TXEND.request	YES	
7.3.5.8	PHY-TXEND.confirm	YES	
7.3.5.9	PHY-CCARESET.request	NO	
7.3.5.10	PHY-CCARESET.confirm	NO	
7.3.5.11	PHY-CCA.indication	NO	
7.3.5.12	PHY-RXSTART.indication	YES	
7.3.5.13	PHY-RXEND.indication	YES	
7.3.5.14	PHY-CONFIG.request	YES	
7.3.5.15	PHY-CONFIG.confirm	YES	
7.4	PHY management	YES	
14	Frequency-Hopping spread spectrum (FHSS) PHY specification for the 2.4GHz industrial, scientific, and medical (ISM) band	YES	
15	Infrared (IR) PHY specification	NO	
16	DSSS PHY specification for the 2.4GHz band designated for ISM applications	YES	
17	High Rate direct sequence spread spectrum (HR/DSSS) PHY specification	YES	
18	Orthogonal frequency division multiplex (OFDM) PHY specification	Partial	2,4 GHz
19	Extended rate PHY (ERP) specification	Partial	2,4 GHz
20	High Throughput (HT) PHY specification	Partial	2,4 GHz

7.3 Additional requirements

7.3.1 General

A device compliant with this PAS shall operate in the license-free 2,4 GHz band using modulations of IEEE STD 802.11-2012. The frequency band, channels, transmission power, and the data rate are defined specifically.

7.3.2 Frequency band

WIA-FA devices operate in the license-free 2,4 GHz band. Different counties define different 2,4 GHz frequency band (see IEEE STD 802.11-2012 PHY). For example, the 2,4 GHz frequency band in China/USA/Europe is defined from 2,4 GHz to 2,4835 GHz, and the 2,4 GHz frequency band in Japan is defined from 2,471 GHz to 2,497 GHz.

7.3.3 Channel bitmap

WIA-FA devices use BitMap (see 6.7.1.2.1) to describe the usable channels. The BitMap is defined by 3 octets to indicate modulation modes and channel indices. The format of the BitMap is shown in Figure 21.

Bits: 0 to 3	Bits: 4 to 17	Bits: 18 to 23
Modulation modes	Channel states	Reserved

Figure 21 – BitMap format

BitMap is composed of the following fields.

– Modulation modes: length is 4 bits. See Table 30 for detailed coding of modulation modes.

– Channel states: Bits 4 to 17 indicate channels in different modulation modes (see Table 31). Bit 4 indicates the usable state of Channel 1; the value of 1 indicates Channel 1 is usable and the value of 0 indicates Channel 1 is not usable. Bits 5 to 17 indicated the usable states of Channel 2 to Channel 14 are same as Bit 4.

The coding method of Modulation modes in Figure 21 according to IEEE STD 802.11-2012 is shown in Table 30.

Table 30 – Coding of Modulation modes

Bit 3 to Bit 0	Modulation
0000	FHSS
0001	DSSS
0010	HR/DSSS
0011	OFDM
0100	ERP-DSSS
0101	ERP-CCK
0110	ERP-OFDM
0111	ERP-PBCC
1000	DSSS-OFDM
1001	No-HT mode
1010	Mix mode
1011	HT mode

The available channels are shown in Table 31.

Table 31 – Channel indices

Modulation		Supported channels
FHSS		
DSSS		Channels indexed from 1 to 14
HR/DSSS		Channels indexed from 1 to 14
OFDM		Not defined, maximum 14 channels
ERP	ERP-DSSS	Channels indexed from 1 to 14
	ERP-CCK	
	ERP-OFDM	
	ERP-PBCC	
	DSSS-OFDM	
HT	No-HT model	Channels indexed from 1 to 14 for 20 MHzchannel spacing; Channels indexed from 1 to 9 or 5 to 13 for 40 MHz channel spacing
	Mix mode	
	HT mode	

7.3.4 Transmission power

Transmit power shall be the Equivalent Isotropic Radiated Power (EIRP) of the device. WIA-FA devices shall provide a nominal EIRP of +10 dBm (10 mW) ± 3dBm. The maximum radiated power level shall not exceed the regulatory requirements that apply where the device is deployed.

7.3.5 Data rate

A WIA-FA device should support the data rates defined on the 2,4 GHz band.

The data rates support by WIA-FA devices are shown in Table 32.

Table 32 – Data rate

Modulation		Data rate (in Mbps)
FHSS		1/1.5/2/2.5/3/3.5/4/4.5
DSSS		1/2
HR/DSSS		1/2/5.5/11
OFDM		1.5/2.25/3/4.5/6/9/12/13.5 for 5 MHz channel spacing (support of 1,5, 3, and 6 Mb/s data rates is mandatory)
ERP	ERP-DSSS	1/2
	ERP-CCK	5.5/11
	ERP-OFDM	6/9/12/18/24/36/48/54
	ERP-PBCC	5.5/11/22/33
	DSSS-OFDM	6/9/12/18/24/36/48/54
HT	No-HT model	Support rates of ERP PHY
	Mix mode	Not defined
	HT mode	Unsupported

8 Data Link Layer

8.1 General

The WIA-FA Data Link Layer (DLL) is designed to guarantee real-time, reliable and secure communication between WIA-FA field devices and access devices. WIA-FA DLL includes:

– DLL data transport functions: adopting TDMA mechanism based on superframe, to avoid transmission collision between messages, and ensure the reliability and real-time performance of transmission; supporting frame aggregation/disaggregation.

– DLL management functions: defining device joining, leaving, time synchronization, remote attribute get/ configuration, etc.

8.1.1 Protocol architecture

WIA-FA DLL protocol architecture is shown in Figure 22. WIA-FA DLL provides service interfaces for AL. DLL includes DLL data entity (DLDE) and DLL management entity (DLME). DLDE is responsible for providing data service interface DLDE-SAP; DLME provides management interface for joining, leaving, time synchronization, configuring parameters and monitoring running status of DLL, etc.

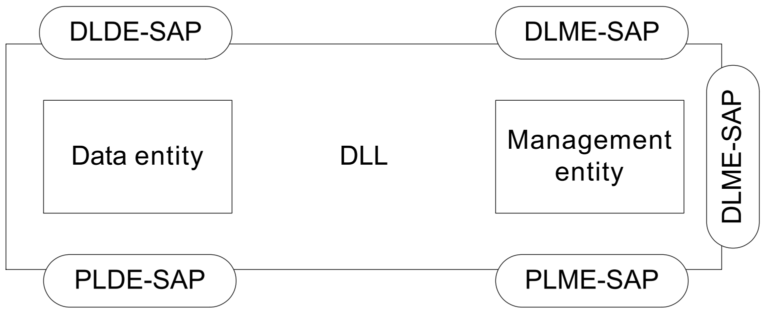

Figure 22 – WIA-FA DLL protocol architecture

8.1.2 WIA-FA superframe

WIA-FA superframe adopts TDMA access mechanism to realize reliability and real-time performance of data transmission. In the sight of time, WIA-FA superframe consists of timeslots and the structure of a timeslot is shown in Figure 23. The length of a timeslot is configurable, and every timeslot is only used for one frame's transmission. In the sight of communication source, WIA-FA superframe consists of several links, and every link is specified by a timeslot and a channel.

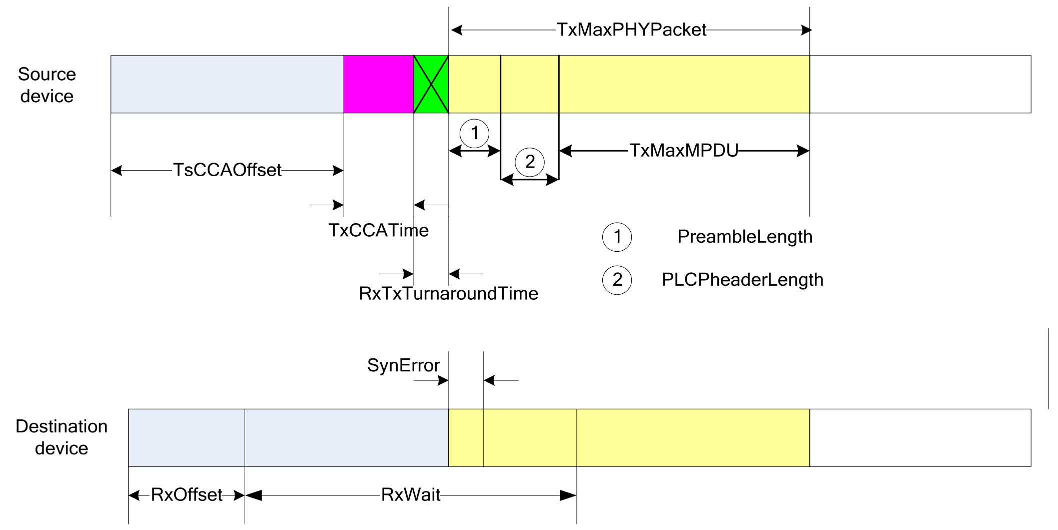

Figure 23 – The template of timeslot structure

The parameters of timeslot template are defined in Table 33.

Table 33 – Parameters of timeslot template

Parameter name	Description
TsCCAOffset	Time from the beginning of a timeslot to the beginning of implementing CCA (in µs)
TsCCATime	Time of implementing CCA (8 symbols)
RxTxTurnaroundTime	Maximum switch time of Rx to Tx transition
PreambleLength	Transmission time of physical layer preamble
PLCPheaderLength	Transmission time of PLCP head
TxMaxDPDU	Transmission time of the longest DLPDU
TxMaxPHYPacket	Transmission time of the longest physical layer frame; the value is PreambleLength+PLCPheaderLength+ TxMaxMPDU (in µs)
RxOffset	Time from the beginning of a timeslot to the beginning of transceiver sensing
RxWait	Shortest time of destination device waiting to start frame transmission, which depends on time shift.
SynError	Time difference from the actual start time of a frame to the ideal time, i.e. the time synchronization difference between a destination device and a source device (in µs)

After WIA-FA network initialized, the gateway device firstly maintains a default superframe. As shown in Figure 24, the default superframe consists of beacon timeslot, management timeslots, and data timeslots. The SuperframeList attribute (see Table 16 in 6.7.1.2.2) of the default superframe is set as follows:

– SuperframeID is set to 0;

– The length of the default superframe is set to 50ms in WIA-FA network;

– ActiveFlag and ActiveSlot are set to 0; ActiveFlag and ActiveSlot can be modified by users.

The structure of the default superframe is broadcast by beacon, and beacon frame format is seen in 8.4.6.

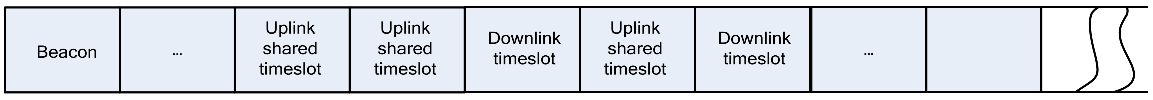

Figure 24 – WIA-FA default superframe

The functions of different kinds of timeslots in default superframe are shown as follows:

– Access device broadcasts beacon at the allocated beacon timeslot in default superframe (see 8.1.2) for field device joining the network (see 8.3.3.5).
– Field device and gateway device use the uplink shared timeslots and downlink timeslots defined in the beacon frame (see 8.4.6) to send join request and join response (see 8.3.4) respectively.
– Gateway device uses the downlink timeslots to configure field devices.
– Gateway device uses the downlink timeslot to invoke remote attribute configuration services to allocate communication resources for field devices.

After a field device joins the network, NM could configure multiple superframes for it. The number of superframes and the length of these superframes are determined by report cycles and data update rates of UAOs in field devices. Same data update rate/report cycle uses the same superframe. That is if device UAO has N data update rates and report cycles, N superframes should be configured. For example, if the data update rates of UAOs are 1, 2, and 16, the report cycles are 2 and 32, the field device should be configured 4 superframes. Gateway device maintains all devices' superframes in the network. The number of superframe and the superframe structure in an access device is the same as that of managed field devices. The structure of a configured superframe is shown in Figure 25.

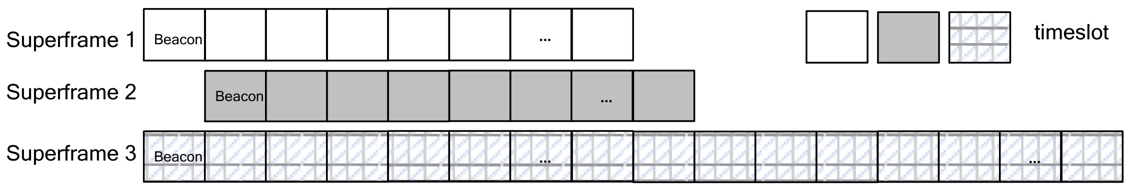

Figure 25 – WIA-FA superframe

WIA-FA network devices adopt TDMA access mechanism to communicate at different channels. As shown in Figure 26, WIA-FA network devices use multiple channels to communicate.

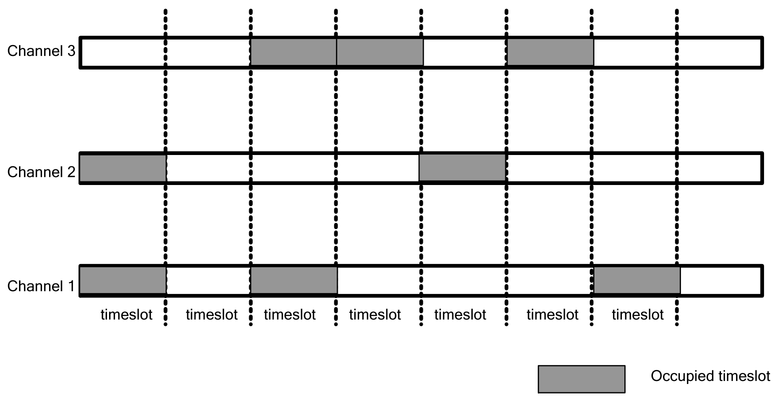

Figure 26 – The example of WIA-FA devices multi-channel communication

8.1.3 Communication based on multiple access devices

8.1.3.1 Beacon communication based on multiple access devices

WIA-FA network allows having multiple access devices. Multiple access devices connect with the gateway device by wires (see Clause 9). NM divides the access devices into multiple sets according to the current available channels, and assigns a channel for every set to send beacon. Access devices in every set could be divided into several teams to send beacon, and the team count is indicated by ADTeamNum. The transmission range of all access devices in one team should cover maximal working field devices at one channel.

NM in the gateway device distributes a unique AdID for each access device (see 9.2). Beacon frames in the WIA-FA network is broadcast by access devices. The NM adopts a grouping strategy for broadcasting beacons, which includes the following steps:

- The default superframe is divided into ADTeamNum segments for ADTeamNum teams of access devices;
- Allocating beacon timeslots from the beginning of each segment for each team. The number of allocated beacon timeslots in one segment equals to the number of access devices in one team;
- Calculating the beacon timeslot number for each access device in a team as

$$(\text{SuperframeLength}/\text{ADTeamNum}) \times \text{TeamID} + \text{InTeamID} \text{ ,}$$

where

SuperframeLength is the length of the default superframe;

TeamID is the team identifier, indexed from 0;

InTeamID is the AD identifier within a team, indexed from 0.

Each field device can realize time synchronization if it receives a beacon from any access device during SuperframeLength. The time precision is improved by multiple time synchronizations within multiple segments.

An example of a TDMA superframe for an access device is shown in Figure 27a). In Figure 27a), the superframe length is 30 timeslots and the timeslot 0 is used for broadcast beacon; Figure 27b) gives an example of a superframe for multiple access devices. 6 access devices are divided into 2 teams. AD11, AD12, and AD13 in one team, and AD21, AD22, and AD23 are in another one team. The superframe in Figure 27b) is divided into two segments and the first three timeslots in each segment are used for broadcasting beacons.

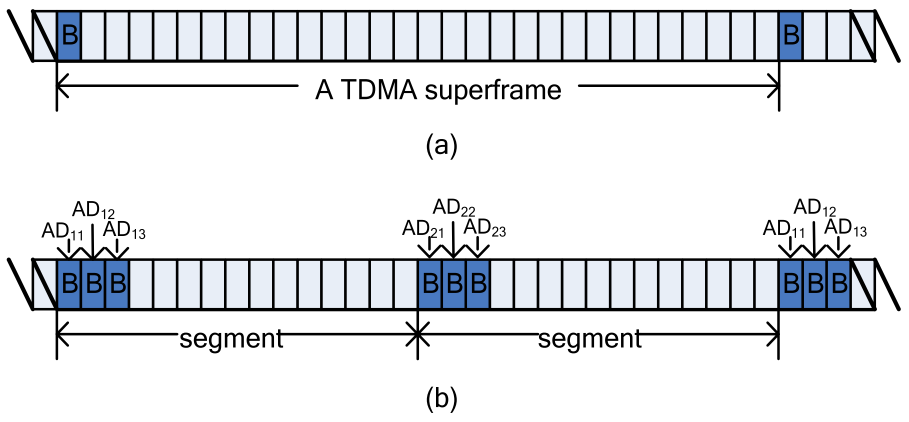

Figure 27 – An example of beacon communication based on multiple ADs

8.1.3.2 Other frame communication based on multiple access devices

When a field device sends a frame according to the LinkList attributes (see Table 17) distributed by NM, the gateway device may receive same frames from multiple access devices. The gateway device filters duplicate frames through the sequence number (see 8.4.1).

When the gateway device sends a frame to a field device, the gateway device should choose an access device to forward the frame by using pre-allocated link (specific selection algorithm in NM is beyond the scope of this PAS).

8.1.4 Time synchronization

The internal clock of the gateway device is set as time source in WIA-FA network. According to IEC 61588, all access devices keep strict time synchronization with the gateway device by wires. WIA-FA network supports one-way time synchronization and two-way synchronization.

– If TwoWayTimeSyn (see 6.7.1.2.1) is set to 0, field devices implement the one-way time synchronization with access device.

– If TwoWayTimeSyn (see 6.7.1.2.1) is set to 1, field devices implement the one-way time synchronization with access device before joining WIA-FA network, implement two-way time synchronization in the first superframe after joining the network, and continue one-way time synchronization after two-way time synchronization. The TxDelay (see 6.7.1.2.1) is recorded during the two-way time synchronization.

For the one-way time synchronization, access devices send beacon frames periodically. After receiving a beacon frame, a field device corrects its local time value according to the timestamp value in a beacon frame to keep the time synchronization of the whole network. The process of the one-way time synchronization is shown in Figure 28. See 8.4.6 for the format of a beacon frame.

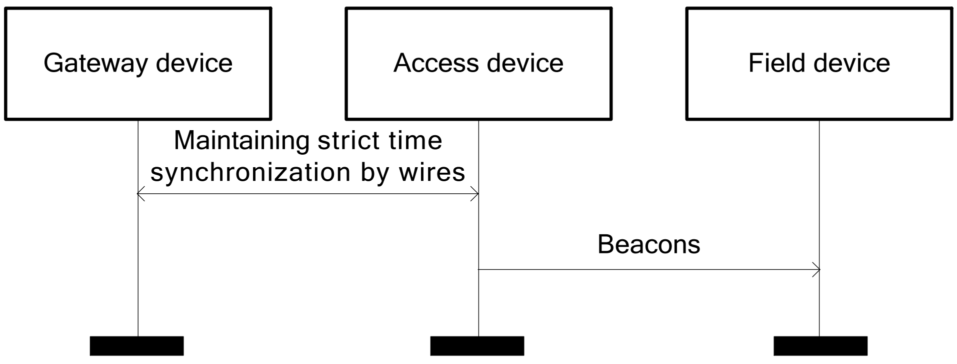

Figure 28 – Process of one-way time synchronization

For the two-way time synchronization, shown in Figure 29, a field device sends a time synchronization request frame (see 8.4.12) to an access device in uplink shared timeslot after receiving a beacon frame. An access device sends the time synchronization response frame (see 8.4.13) to the field device in management timeslot. The payload of the time synchronization response frame contains FieldDeviceTimeValue and ReceiveTimeValue. The field device implements the two-way time synchronization and calculates TxDelay according to Absolute time value in Beacon frame, received time of this beacon, FieldDeviceTimeValue, and ReceiveTimeValue. TxDelay is calculated as:

TxDelay= ((received time of this beacon – Absolute time value in Beacon frame)+(ReceiveTimeValue- FieldDeviceTimeValue))/2

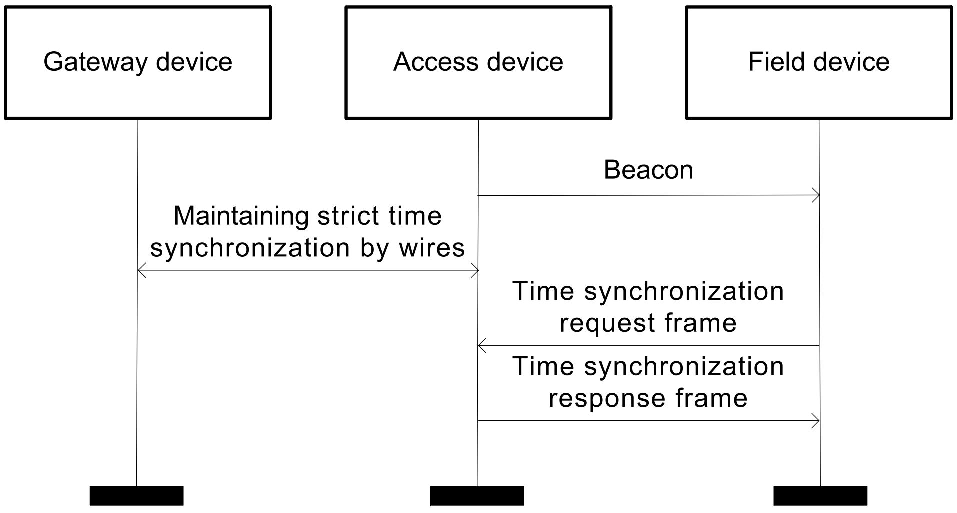
Gateway device
Access device
Field device
Beacon
Maintaining strict time synchronization by wires
Time synchronization request frame
Time synchronization response frame

Figure 29 – Process of two-way time synchronization

8.1.5 Aggregation/Disaggregation

The DLL of WIA-FA supports frame aggregation/disaggregation mechanism to reduce the number of transmitted frames.

The aggregation shall apply to RT1 and RT3 data. The aggregated frames should be in same priority, i.e., RT1 or RT3.

Frame aggregation/disaggregation is a function of DLL. Access devices support frame aggregation and field device supports disaggregation. When the DLL of an access device needs to send N data frames to field devices, the access device shall figure out the length of frame after aggregation. If it is less than the MaxPayloadLength of aggregated frame (see 6.7.1.2.1), the DLL of access device aggregates those data frames. After receiving aggregated data frame, each field device disaggregates the aggregated frame and gets its own frame. The aggregation/disaggregation mechanism can reduce the number of frames from access devices to field devices and can improve the network capacity.

Frame aggregation is accomplished by DLL in an access device. The configuration process of aggregation function is as follows:

– Host computer reads the AGGSupportFlag (see 6.7.1.2.1) from field devices and access devices to determine whether they support frame aggregation/disaggregation function. If the AGGSupportFlag of both field devices and access devices are 1, host computer continues the following configuration process. Otherwise, AGGEnableFlag (see 6.7.1.2.1) in all the WIA-FA devices are set to 0.
– If the aggregation/disaggregation function is enabled, NM shall set AGGEnableFlag in all WIA-FA devices to 1.

The frame aggregation/disaggregation function of an access device is performed as follows:

– If the value of AGGEnableFlag is 0, the access device shall not enable the frame aggregation/disaggregation function.
– If the value of AGGEnableFlag is 1, the access device shall enable the frame aggregation/disaggregation function. The access devices aggregate multiple frames for multiple field devices according to the format of the aggregated frame shown in Figure 30. The length of aggregation frame shall be less than the max length MaxPayloadLength (see 6.7.1.2.1). DLL of access device sets Frame Type (see 8.4.1) to aggregation frame, and uses broadcast timeslot (receiving timeslot in the corresponding field devices) which network manager pre-allocated for access devices to send aggregation frames.

	The first frame			**…**	**The nth frame**		
1 octet	1/2 octets	2 octets	Variable Length	…	1/2 octets	2 octets	Variable Length
Aggregated number	Field device address	Data length	Data	…	Field device address	Data length	Data

Figure 30 – Aggregation frame payload format

Each field in Figure 30 is defined as follows:

– Aggregation number: length is 1 octet, indicates the number of frames that is aggregated to send to field devices.

– Field device address: length is 1 or 2 octets, indicates the destination address of following aggregated data.

– Data length: length is 2 octets, represents the length of data that is sent to a field device.

– Data: variable length, represents the data that is sent to a field device.

The disaggregation function is set and used as follows:

– If AGGSupportFlag of field device and access device is 1 and network manager set AGGEnableFlag of field device and access device to 1, then aggregation function and disaggregation function enabled at the same time, i.e. access device enabled frame aggregation function meanwhile field device enabled disaggregation function.

– Whether field device disaggregates depends on the frame type of the received frame. If Frame Type (see 8.4.1) indicates aggregation frame, field device needs to disaggregate according to the format of aggregation frame (see Figure 30).

8.1.6 Retransmission

8.1.6.1 Retransmission modes

WIA-FA supports the following retransmission modes:

a) NACK-based retransmission mode: when a field device sends data to the gateway device periodically, it adopts the retransmission mode based on NACK.

b) Multi-unicast retransmission mode: when the gateway device sends non-aggregated data to a field device periodically, it unicasts the same frame multiple times to this field device.

c) Multi-broadcast retransmission mode: when the gateway device periodically sends aggregation frame, it broadcasts the same aggregation frame multiple times to field devices.

d) GACK-based timeslot backoff mode: when a field device sends a non-periodic data frame or management frame (e.g., remote attribute get, remote attribute set, two-way time synchronization) to the gateway device, it retries by backoff the retransmission timeslot according to GACK.

The non-aggregated broadcast frame from the gateway device to field devices doesn’t need to return ACK, thus no retransmission is needed.

NACK frames and GACK frames should be sent multiple times to ensure the network reliability.

8.1.6.2 NACK-based retransmission mode

The NACK retransmission mode is realized according to the following steps.

– The NM allocates several groups of retransmission timeslots for periodic data exchange between field devices and the gateway device. These periodic data can be data or management frames. The count of groups and the count of timeslots in each group are determined by Lossrate and TargetLossRate (see 6.7.1.2.1).

A method of reserving retransmission timeslots is designed as follows.

The minimum count of retransmission times minRetryTime is calculated as

minRetryTime= log LossrateTargetLossRate.

Then, the retransmission times MaxRetry >=minRetryTime (see 6.7.1.2.1 for MaxRetry).

The count of the nth group retransmission timeslots minRetrySlotNum[n] satisfy

minRetrySlotNum[n]>=FrameCount * TargetLossRaten,

where

FrameCount is the count of periodic frames sent to the gateway device during one superframe.

– A field device firstly uses a pre-allocated timeslot to send a periodic data/management frame to the gateway device.
– The gateway device generates NACK frame (see 8.4.4) in a certain order after he receives multiple periodic data/management frames and broadcasts the NACK frame multiple times. The NACK frame includes the short addresses of the failure field devices, from which the gateway device does not receive periodic data/management frames in scheduled timeslots. The repeating count of NACKs is MaxRetry.
– Field devices parse the received NACKs. If the payload the NACK frame has the address of a field device, this field device shall retransmit its periodic frame by using the retransmission timeslot in the order timeslot indicated by the NACK payload. If the scheduled retransmission timeslots are not enough for field device retransmissions, the field device shall postpone its retransmissions according to next NACK.

Figure 31 is an example of NACK retransmission mode. Supposing that Lossrate is 0,1 and TargetLossRate is 0,01 %, then minRetryTime = 4. Let MaxRetry>= minRetryTime be 4. Each field device firstly uses scheduled timeslots to send periodic data/management frames to the gateway device. The gateway device generates NACK frame and broadcasts 4 times to field devices indicating whether it receives frames from field devices. Field devices retry periodic data/management frames in scheduled retransmission timeslots according to the sequence indicated in NACK frame payload.

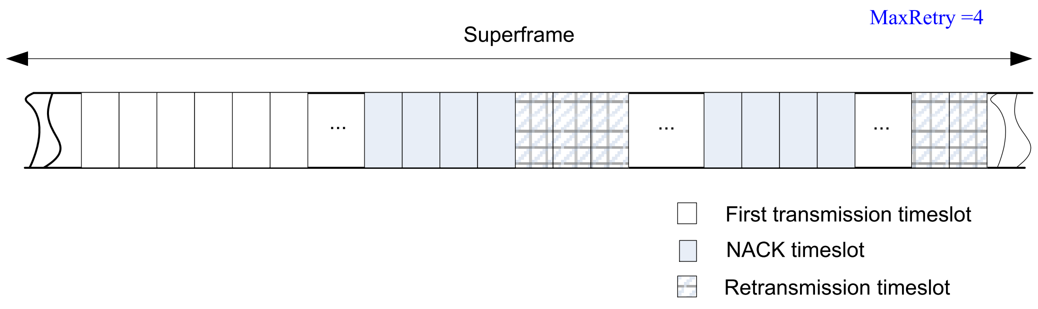

Figure 31 – Example of retransmission mode based on NACK

8.1.6.3 Multi-unicast retransmission mode

The multi-unicast retransmission mode is used by the gateway device to send non-aggregated and periodic data/management frames. The gateway device sends the non-aggregated and periodic data/management frames to a field device by multiple times until MaxRetry (see 6.7.1.2.1).

An example of multi-unicast retransmission mode is shown in Figure 32.

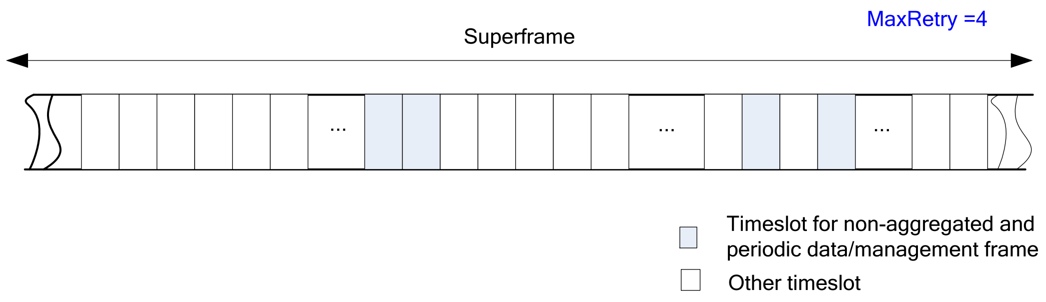
MaxRetry =4
Superframe
...
...
...
Timeslot for non-aggregated and periodic data/management frame
Other timeslot

Figure 32 – Example of multi-unicast retransmission mode

8.1.6.4 Multi-broadcast retransmission mode

In a WIA-FA network supporting aggregation/disaggregation mechanism, the gateway device broadcasts an aggregated frame to field devices by multiple times until MaxRetry.

An example of multi-broadcast retransmission mode is shown in Figure 33.

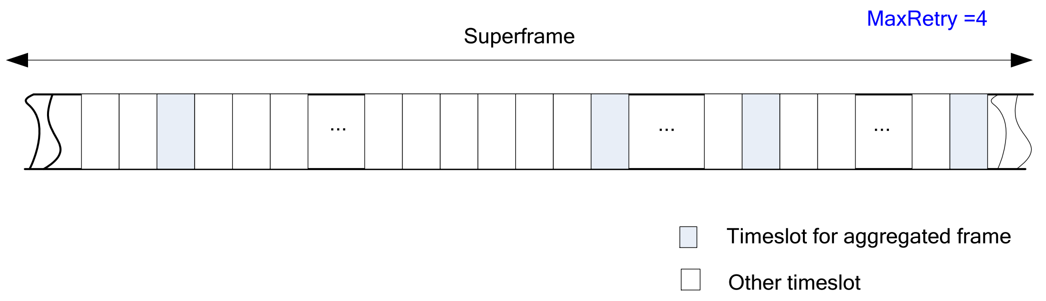

Figure 33 – Example of multi-broadcast retransmission mode

8.1.6.5 GACK-based timeslot backoff mode

Field devices utilize the timeslot backoff retransmission mode to retry aperiodic data/management frames (remote get attribute frame, remote set attribute frame, time synchronization frame, etc.) to the gateway device based on NACKs (see 8.4.5).

After the gateway device receives aperiodic data/management frames from multiple field devices, it generates a GACK frame (see 8.4.5) according to the addresses of these field devices. The generated GACK is broadcast by multiple times, which is same as that of NACK-based retransmission mode (see 8.1.6.2). If a field device does not receive a NACK frame or the received NACK frame does not include its address, the field device retries the related aperiodic data/management frame by using the timeslot backoff method to compete retransmission timeslots.

The retransmitting timeslots (see LinkList in 6.7.1.2.2) in each superframe shall be used by field devices to retry aperiodic data/management frames. The field devices set links as receiving links in GACK broadcast timeslot and prepare to receive the GACK frame. If a field device does not receive a NACK frame or the received NACK frame does not include its address, it completes the retransmitting timeslots for sending aperiodic data/management frames. If the competition is failure, the field device shall delay its retransmission to next retransmitting timeslot until MaxRetry (see 6.7.1.2.1).

An example of GACK-based timeslot backoff mode is shown in Figure 34.

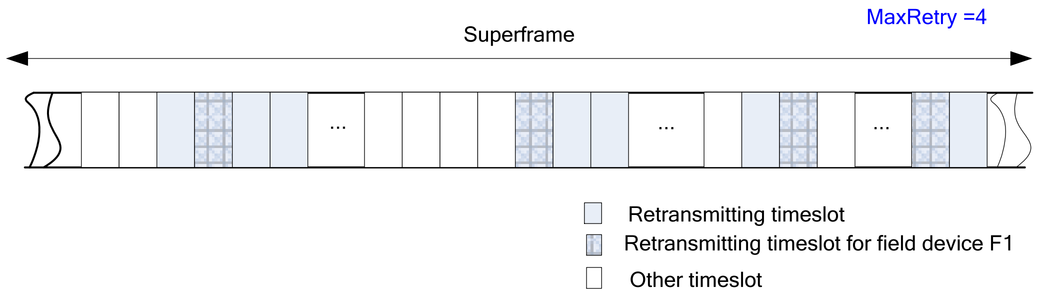

Figure 34 – Example of GACK-based timeslot backoff mode

8.2 Data link sub-layer data services

8.2.1 General

The data link layer data entity service access point (DLDE-SAP) supports the point-to-point transmission between an access device and a field device. The DLL data services primitive includes DLDE-DATA.request, DLDE-DATA.confirm, and DLDE-DATA.indication.

8.2.2 DLDE-DATA.request primitive

Application sub-layer invokes DLDE-DATA.request primitive to send data.

The semantics of DLDE-DATA.request primitive are as follows:

```
DLDE-DATA.request(
                          DstAddr,
                          VCR_ID,
                          DataType,
                          Priority,
                          PayloadLength,
                          Payload
                          )
```

Table 34 specifies the parameters of DLDE-DATA.request primitive.

Table 34 – DLDE-DATA.request primitive parameters

Name	Data type	Valid range	Description
DstAddr	Unsigned16	0 to 65 535	The short address of destination device
VCR_ID	Unsigned16	0 ～65535	VCR identifier for data. VCR_ID is invalid only if DataType =0

DataType	Unsigned8	0 to 255	Data type:
			0 = DATA;
			1 = NACK;
			2 = GACK;
			others are reserved

Priority	Unsigned8	0 to 255	The priority of the payload,
			0 = RT0;
			1 = RT1;
			2 = RT2;
			3 = RT3;
			4 = NRT;
			others are reserved.

PayloadLength	Unsigned16	0 to 65 535	Denoting the length of payload (the unit is octet)
Payload	Octetstring		Payload

8.2.3 DLDE-DATA.indication primitive

DLDE-DATA.indication primitive is used to report data received to the application sub-layer.

The semantics of DLDE-DATA.indication primitive are as follows:

```
DLDE-DATA.indication(
                    SrcAddr,
                    DataType,
                    PayloadLength,
                    Payload
                    )
```

Table 35 specifies the parameters of DLDE-DATA.indication.

Table 35 – DLDE-DATA.indication primitive parameters

Name	Data type	Valid range	Description
SrcAddr	Unsigned16	0 to 65 535	The short address of source device

DataType	Unsigned8	0 to 255	Data type:
			0 = DATA;
			1 = NACK;
			2 = GACK;
			others are reserved

PayloadLength	Unsigned16	0 to 65 535	Denoting the length of Payload (in octet)
Payload	Octetstring		Payload

8.2.4 Time sequence of DLL data service

Figure 35, Figure 36, and Figure 37 give the basic process of data frame transmission, reception, and acknowledgement.

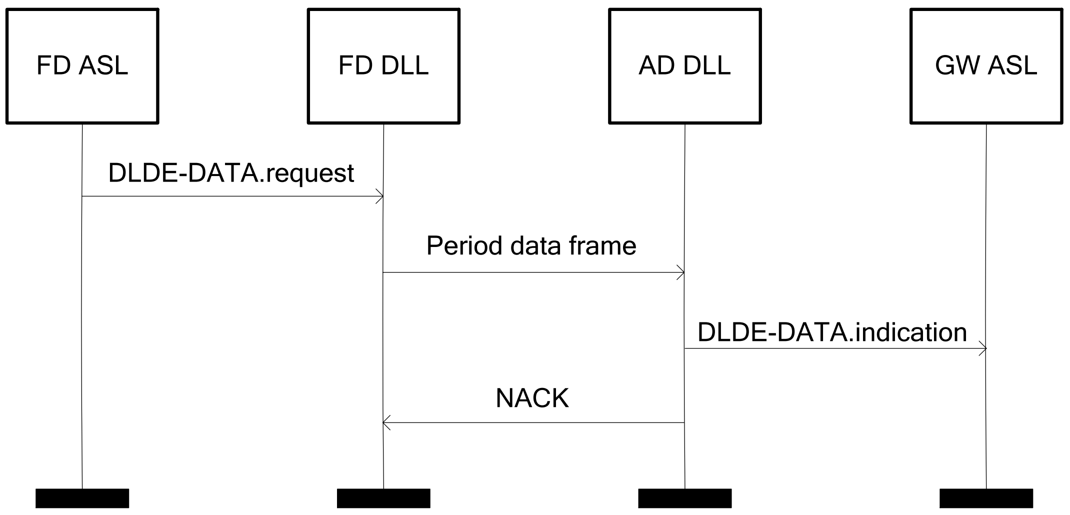

Figure 35 – Time sequence of period data service from FD to GW

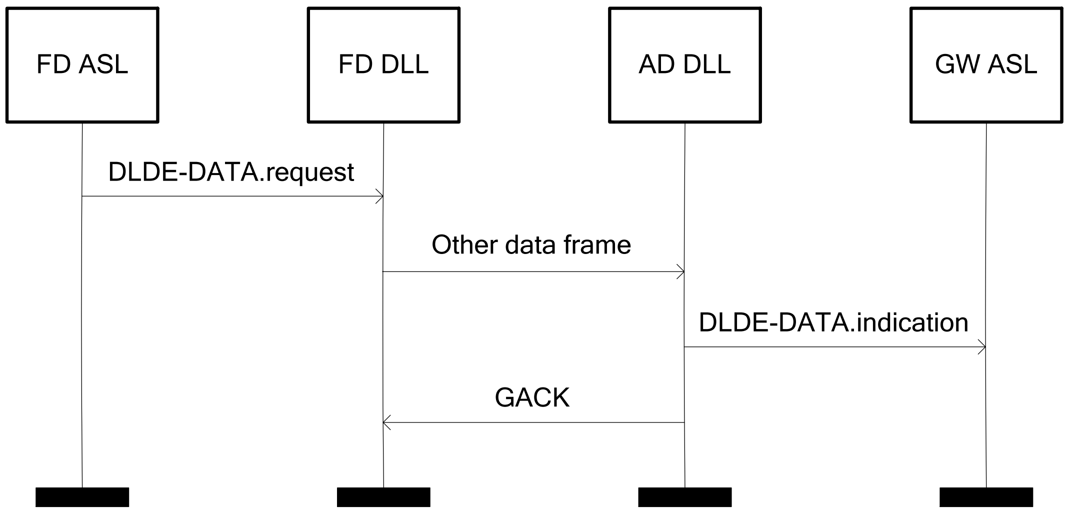

Figure 36 – Time sequence of other data service from FD to GW

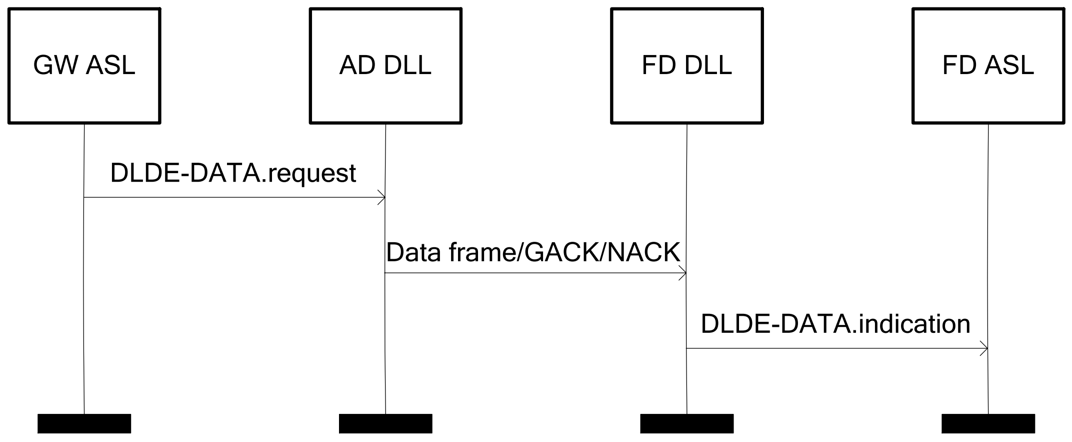

Figure 37 – Time sequence of data service from GW to FD

8.3 Data link sub-layer management services

8.3.1 General

The data link layer management entity service access point (DLME-SAP), defines the way that application layer passes management commands to the data link layer. Table 36 summarise all the management services.

All Request services are invoked by DMAP to DLL to generate a request command frame; Indication services are used by DLL to DMAP to report receiving a command frame; Response services are invoked by DMAP to DLL to generate a response frame; and Confirm services are used by DLL to DMAP to return the transmission status of a request command frame.

Table 36 – Management services

Service name	Request	Indication	Response	Confirm
DLME-DISCOVERY	8.3.2.1			8.3.7.2
DLME-TIME-SYN	8.3.3.1	8.3.3.2	8.3.3.3	0
DLME-JOIN	8.3.4.1	8.3.4.2	8.3.8.2	8.3.9.2
DLME-DEVICE-STATUS	8.3.5.1	8.3.5.2		8.3.5.3
DLME-CHANNEL-CONDITION	8.3.6.1	8.3.4.3		8.3.6.3
DLME-INFO-GET	8.3.7.1	8.3.7.3	8.3.8.3	8.3.7.4
DLME-INFO-SET	8.3.8.1	8.3.2.2	8.3.4.4	8.3.8.4
DLME-LEAVE	8.3.9.1	8.3.9.2		8.3.9.3

8.3.2 Network discovery services

8.3.2.1 DLME-DISCOVERY.request

DLME-DISCOVERY.request is used to request a device to scan channels.

```
DLME-DISCOVERY.request(
                    ScanChannels
                    )
```

Table 37 specifies the parameters for DLME-DISCOVERY.request.

Table 37 – DLME-DISCOVERY.request parameters

Parameter	Data type	Valid range	Description
ScanChannels	Unsigned8	Bitmap (see 6.7.1.2.1)	IEEE STD 802.11-2012 14 physical channels available

8.3.2.2 DLME-DISCOVERY.confirm

DLME-DISCOVERY.confirm is used to respond to DLME-DISCOVERY.request.

```
DLME-DISCOVERY.confirm(
                        Status,
                        BeaconCount,
                        SuperframeLength,
                        TimeslotDuration,
                        FirstSharedTimeslotNum,
                        SharedTimeslotCount,
                        AbsoluteTimeValue,
                        BeaconDescription
                        )
```

Table 38 specifies the parameters for DLME-DISCOVERY.confirm.

Table 38 – DLME-DISCOVERY.confirm parameters

Parameter	Data type	Valid range	Description

Status	Unsigned8	0 to 255	Scan results:
			0 = SUCCESS;
			1 = NO_BEACON;
			Others are reserved.

BeaconCount	Unsigned8	0 to 255	The number of beacons discovered
SuperframeLength	Unsigned16	0 to 65 535	specifies the length of default superframe in timeslot (see 8.1.2)
TimeslotDuration	Unsigned16	0 to 65 535	The timeslot length set by network (see TimeSlotDuration in 6.7.1.2.1)
FirstSharedTimeslotNumber	Unsigned16	0 to 65 535	The shared timeslots in superframe
ShareTimeslotNumber	Unsigned8	0 to 255	The total number of shared timeslots (see 8.4.6)
AbsoluteTimeValue	TimeData	0 to (2^{64}–1)	The absolute time value sending the beacon (see TimeValue in 6.7.1.2.1)
BeaconDescription	List of BeaconDescription_ Struct structure		See Table 39

Table 39 – BeaconDescription_Struct parameters

Name	Data type	Valid range	Description
ChannelIndex	Unsigned8	0 to 255	The channel of receiving Beacon (see BitMap in 6.7.1.2.1)
BeaconRelativeTimeslotNum	Unsigned16	0 to 65 535	The timeslot sending beacon
ED	Unsigned8	0 to 255	The energy level of a received Beacon

8.3.2.3 Network discovery process

The process of the network discovery is shown in Figure 38.

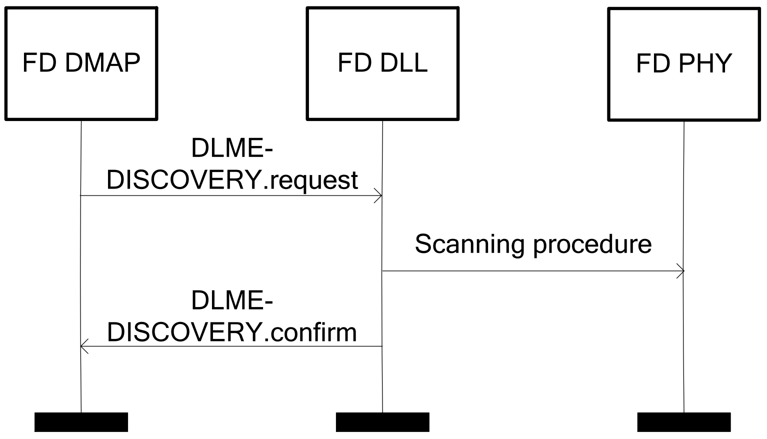

Figure 38 – Network discovery process

The field device DMAP invokes DLME-DISCOVERY.request to request the physical layer invokes the scanning procedure (see 10.1.4.3.3 in IEEE 802.11-2012). The DLL reports scanning results to DMAP by DLDE- DISCOVERY.confirm after scanning.

8.3.3 Time synchronization services

8.3.3.1 DLME-TIME-SYN.request

DLME-TIME-SYN.request is used to send two-way time synchronization command frame requested by the DMAP.

```
DLME-TIME-SYN.request(
                                  )
```

DLME-TIME-SYN.request parameters are null, which request physical layer to write the absolute time value in the time synchronization request command frame.

8.3.3.2 DLME-TIME-SYN.indication

DLME-TIME-SYN.indication is used to inform the DMAP that the time synchronization request command frame has been successfully received.

```
DLME-TIME-SYN.indication(

                                  SrcAddr,
                                  FieldDeviceTimeValue
                                  )
```

Table 40 specifies the parameters for DLME-TIME-SYN.indication.

Table 40 – DLME-TIME-SYN.indication parameters

Parameter	Data type	Valid range	Description
SrcAddr	Unsigned16	0 to 65 535	The short address of the field device (see DeviceShortAddress in 6.7.1.2.1)
FieldDeviceTimeValue	TimeData	0 to (2^{64}-1)	The timestamp when the field device sends time synchronization request frame, in microseconds, see 8.4.12

8.3.3.3 DLME-TIME-SYN.response

DLME-TIME-SYN.response is the response of DLME-TIME-SYN.indication.

```
DLME-TIME-SYN.response(
                        DstAddr,
                        FieldDeviceTimeValue,
                        ReceiveTimeValue
                        )
```

Table 41 specifies the parameters for DLME-TIME-SYN.response.

Table 41 – DLME-TIME-SYN.response parameters

Parameter	Data type	Valid range	Description
DstAddr	Unsigned16	0 to 65 535	The short address of the field device (see DeviceShortAddress in 6.7.1.2.1)
FieldDeviceTimeValue	TimeData	0 to $(2^{64}-1)$	The timestamp when the field device sends time synchronization request command frame (in microseconds), see 8.4.12
ReceiveTimeValue	TimeData	0 to $(2^{64}-1)$	The timestamp when the access device receives time synchronization request command frame (in microseconds), see 8.4.13

8.3.3.4 DLME-TIME-SYN.confirm

DLME-TIME-SYN.confirm is used to forward the time value that the field device sends the two-way time synchronization frame or the access device receives the time synchronization request frame in the two-way time synchronization frame (see 8.4.12).

```
DLME-TIME-SYN.confirm(
                        Status,
                        FieldDeviceTimeValue,
                        ReceiveTimeValue
                        )
```

Table 42 specifies the parameters for DLME-TIME-SYN.confirm.

Table 42 – DLME-TIME-SYN.confirm parameters

Parameter	Data type	Valid range	Description
Status	Unsigned8	0 to 255	Result of the time synchronization request: 0 = SUCCESS; 1 = OVERTIME (When the overtime >= TwoWayOverTime, the time synchronization is failed); Others are reserved. See 6.7.1.2.1 for TwoWayOverTime.
FieldDeviceTimeValue	TimeData	0 to $(2^{64}-1)$	The timestamp when the field device sends two-way time synchronization request command frame, in microseconds, see 8.4.12
ReceiveTimeValue	TimeData	0 to $(2^{64}-1)$	The timestamp when the access device receives two-way time synchronization request command frame (in microseconds), see 8.4.13

8.3.3.5 Time synchronization process

Time synchronization process is shown in Figure 39.

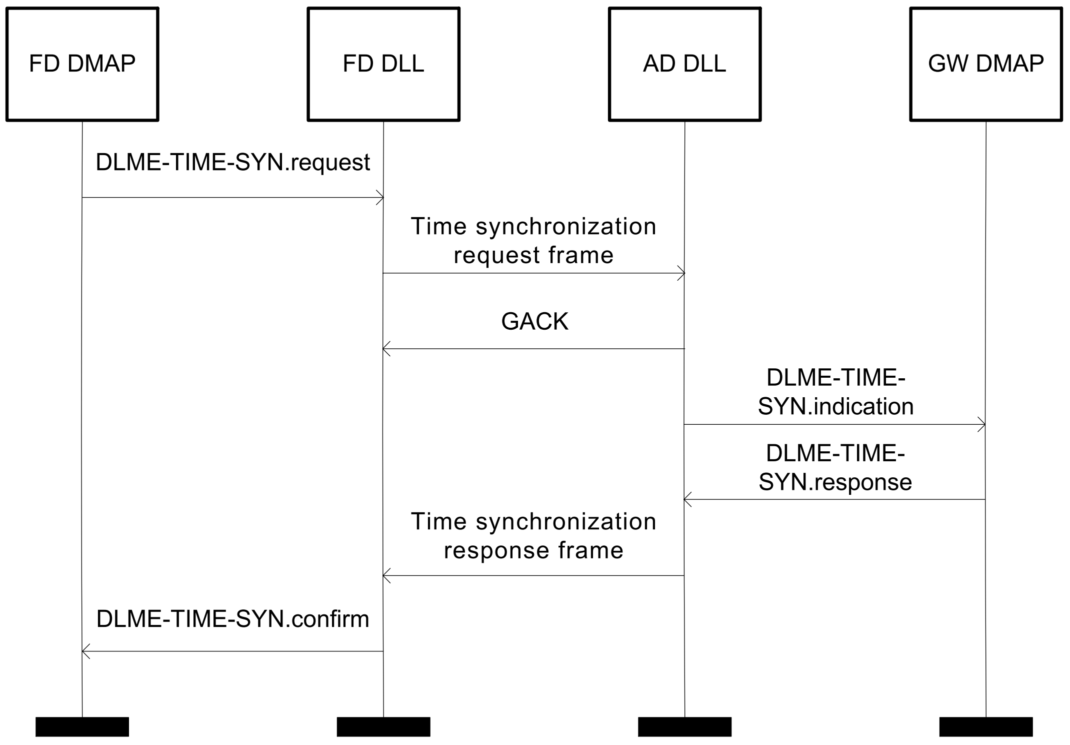

Figure 39 – Time synchronization process

The field device DMAP invokes DLME-TIME-SYN.request in DLL, indicating DLL send time synchronization request command frame; After receiving the time synchronization request command frame, the access device returns GACK, and reports to the gateway device DMAP by DLME-TIME-SYN.indication. The gateway device DMAP invokes DLME-TIME-SYN.response primitive, indicating the access device DLL sending time synchronization response command frame. After receiving the time synchronization response command frame, the field device DLL confirms to DMAP by DLME-TIME-SYN.confirm.

8.3.4 Device joining services

8.3.4.1 DLME-JOIN.request

DLME-JOIN.request is invoked by DMAP of a field device attempt to join WIA-FA network to request DLL generating a join request command frame.

```
DLME-JOIN.request(
                        NetworkID,
                        Channel,
                        PhyAddr,
                        SecMaterial
                        )
```

Table 43 specifies the parameters for DLME-JOIN.request.

Table 43 – DLME-JOIN.request parameters

Parameter	Data type	Valid range	Description
NetworkID	Unsigned8	0 to 255	Network identifier, used for multiple networks coexisting (see NetworkID in 6.7.1.2.1)
Channel	Unsigned8	Bitmap (see 6.7.1.2.1, Table 17)	Channel used for joining, chosen from valid channels supported by PHY
PhyAddr	Unsgned64	0 to $(2^{64}-1)$	Long address of the new device waiting to join (see LongAddress in 6.7.1.2.1)
SecMaterial	Unsgned64	0 to $(2^{64}-1)$	Device security material for authentication (see 11.3). If the SecLevel is 0, this field is NULL.

8.3.4.2 DLME-JOIN.indication

DLME-JOIN.indication is used to inform DMAP of the gateway device that the join request from one device has been successfully received.

```
DLME-JOIN.indication(
                    PhyAddr,
                    SecMaterial
                    )
```

Table 44 specifies the parameters for DLME-JOIN.indication.

Table 44 – DLME-JOIN.indication parameters

Parameter	Data type	Valid range	Description
PhyAddr	Unsigned64	0 to $(2^{64}-1)$	Long address of the new device waiting to join (see LongAddress in 6.7.1.2.1)
SecMaterial	Unsgned64	0 to $(2^{64}-1)$	Device security material for authentication (see 11.3). If the SecLevel is 0, this field is NULL.

8.3.4.3 DLME-JOIN.response

DLME-JOIN.response is the response of DLME-JOIN.indication.

```
DLME-JOIN.response(
                    Status,
                    ShortAddr
                    )
```

Table 45 specifies the parameters for DLME-JOIN.response.

Table 45 – DLME-JOIN.response parameters

Parameter	Data type	Valid range	Description

Status	Unsigned8	0 to 255	Result of join requestjoin request:
			0 = SUCCESS
			1 = NetworkID mismatched;
			2 = Authentication failure;
			3 = Network overload;
			others are reserved.

ShortAddr	Unsigned16	0 to 65 535	Short address allocated by the GW to device waiting to join (see DeviceShortAddress in 6.7.1.2.1), if Status=SUCCESS, this field is valid.

8.3.4.4 DLME-JOIN.confirm

DLME-JOIN.confirm is a response to DLME-JOIN.request.

```
DLME-JOIN.confirm(
                        Status,
                        ShortAddr
                        )
```

Table 46 specifies the parameters for DLME-JOIN.confirm.

Table 46 – DLME-JOIN.confirm parameters

Parameter	Data type	Valid range	Description

Status	Unsigned8	0 to 255	Result of join requestjoin request:
			0 = SUCCESS;
			1 = NetworkID mismatched;
			2 = authentication fail;
			3 = network overload;
			Others are reserved.

ShortAddr	Unsigned16	0 to 65 535	Short address allocated by the GW to device waiting to join (see DeviceShortAddress in 6.7.1.2.1), if Status=SUCCESS, this field is valid.

8.3.4.5 Device join process

Device join process is shown in Figure 40.

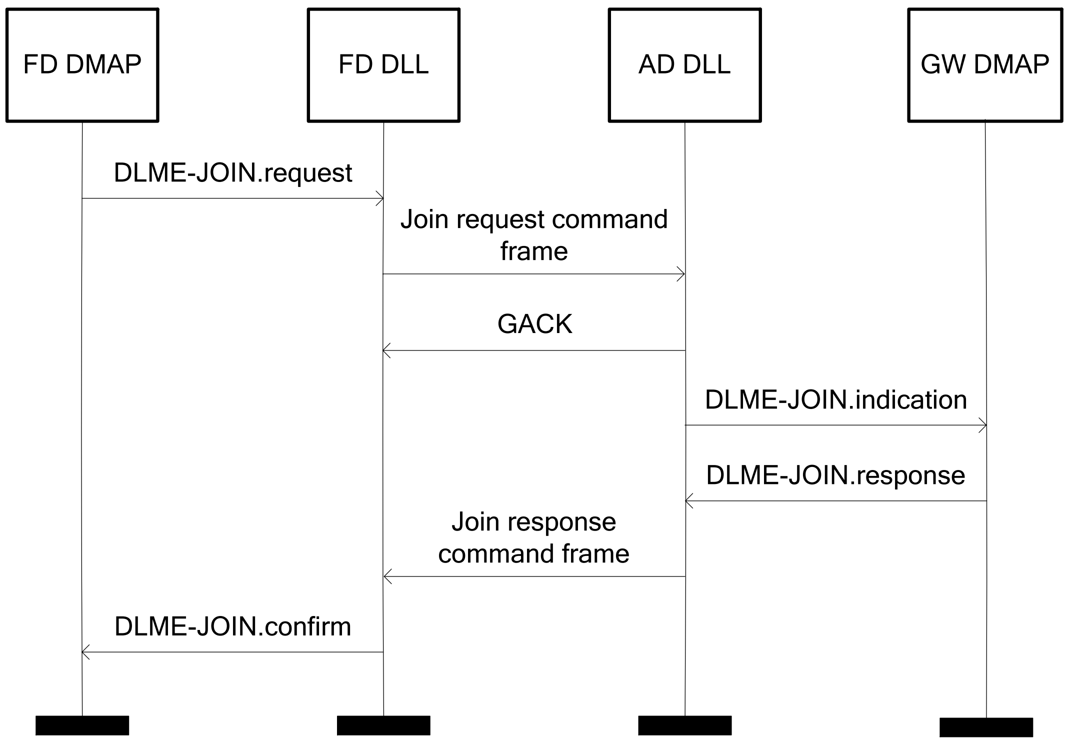

Figure 40 – Device join process

The field device DMAP invokes DLME-JOIN.request primitive in DLL, indicating DLL send join request command frame; After receiving join request command frame, the access device returns GACK, and reports to the gateway device DMAP by DLME-JOIN.indication; The gateway device DMAP invokes DLME-JOIN.response primitive, indicating the access device DLL sending join response frame. After receiving join response command frame, the field device DLL confirms to DMAP by DLME-JOIN.confirm.

8.3.5 Device status report services

8.3.5.1 DLME-DEVICE-STATUS.request

DLME-DEVICE-STATUS.request is used to periodically report the device status to the gateway device by a field device.

```
DLME-DEVICE-STATUS.request(
                              PowerSupplyStatus
                              )
```

Table 47 specifies the parameters for DLME-DEVICE-STATUS.request.

Table 47 – DLME-DEVICE-STATUS.request parameters

Parameter	Data type	Valid range	Description
PowerSupplyStatus	Unsigned8	0 to 255	Information of device electric power, (See PowerSupplyStatus in Device_Struct structure)

8.3.5.2 DLME-DEVICE-STATUS.indication

DLME-DEVICE-STATUS.indication is used to report the receipt of a device condition report command frame to the DMAP.

DLME-DEVICE-STATUS.indication(

PowerSupplyStatus
)

Table 48 specifies the parameters for DLME-DEVICE -STATUS.indication.

Table 48 – DLME-DEVICE -STATUS.indication parameters

Parameter	Data type	Valid range	Description
PowerSupplyStatus	Unsigned8	0 to 255	Information of device electric power (see PowerSupplyStatus in Device_Struct structure)

8.3.5.3 DLME-DEVICE-STATUS.confirm

DLME-DEVICE-STATUS.confirm is used to return the results of DLME-DEVICE-STATUS.request.

DLME- DEVICE-STATUS.confirm(
Status
)

Table 49 specifies the parameters for DLME-DEVICE-STATUS.confirm.

Table 49 – DLME-DEVICE -STATUS.confirm parameters

Parameter	Data type	Valid range	Description

Status	Unsigned8	0 to 255	Result of the device status report request:
			0 = SUCCESS;
			1 = FAILURE;
			others are reserved.

8.3.5.4 Device status report process

Device status report process is shown in Figure 41.

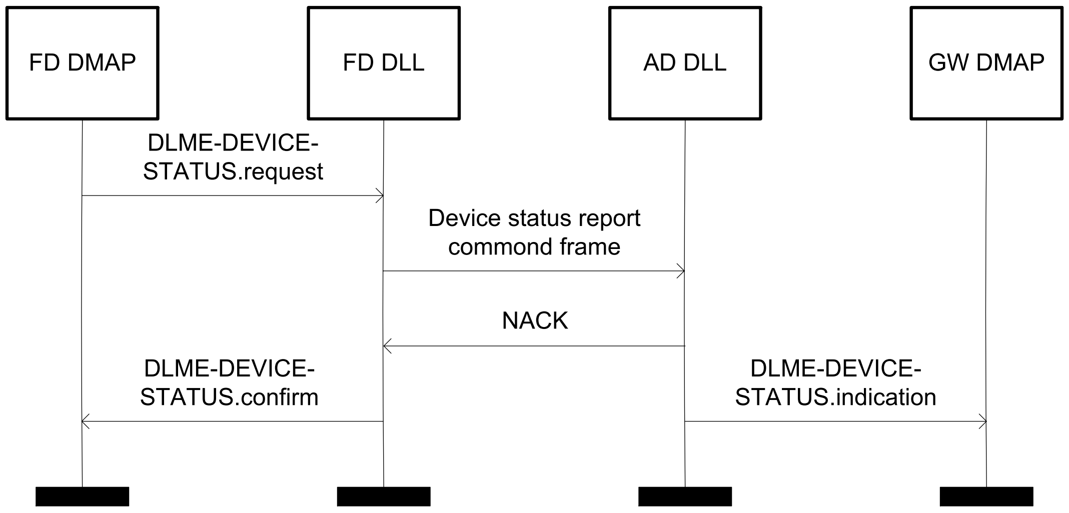
FD DMAP
FD DLL
AD DLL
GW DMAP
DLME-DEVICE-STATUS.request
Device status report commond frame
NACK
DLME-DEVICE-STATUS.confirm
DLME-DEVICE-STATUS.indication

Figure 41 – Device status report process

The field device DMAP invokes DLME-DEVICE-STATUS.request in DLL, indicating DLL send device status report command frame; After receiving device status report command frame, the access device returns NACK, and reports to the gateway device DMAP by DLME-DEVICE-STATUS.indication. The field device DLL confirms to DMAP by DLME-DEVICE-STATUS.confirm after receiving NACK.

8.3.6 Channel condition report services

8.3.6.1 DLME-CHANNEL-CONDITION.request

DLME-CHANNEL-CONDITION.request is used to report the communication channel condition to the gateway device by a field device.

```
DLME-CHANNEL-CONDITION.request(
                                Count,
                                ChannelConditionInfo
                                )
```

Table 50 specifies the parameters for DLME-CHANNEL-CONDITION.request.

Table 50 – DLME-CHANNEL-CONDITION.request parameters

Parameter	Data type	Valid range	Description
Count	Unsigned8	0 to 255	The length of list of ChanCon_Struct
ChannelConditionInfo	List of ChanCon_Struct (see Table 18)		Information of channel condition attributes

8.3.6.2 DLME-CHANNEL-CONDITION.indication

DLME-CHANNEL-CONDITION.indication is used to report the receipt of a channel condition report command frame to the DMAP.

```
DLME-CHANNEL-CONDITION.indication(
                                SrcAddr,
                                Count,
                                ChannelConditionInfo
                                )
```

Table 51 specifies the parameters for DLME-CHANNEL-CONDITION.indication.

Table 51 – DLME-CHANNEL-CONDITION.indication parameters

Parameter	Data type	Valid range	Description
SrcAddr	Unsigned16	00 to 65 535	The short address of the device asking to leave (see DeviceShortAddress in 6.7.1.2.1)
Count	Unsigned8	0 to 255	The length of list of ChanCon_Struct
ChannelConditionInfo	List of ChanCon_Struct structure (see Table 18)		Information of channel condition attributes

8.3.6.3 DLME-CHANNEL-CONDITION.confirm

DLME-CHANNEL-CONDITION.confirm is used to return the results of DLME-CHANNEL-CONDITION.request.

```
DLME-CHANNEL-CONDITION.confirm(
```

Status
)

Table 52 specifies the parameters for DLME-CHANNEL-CONDITION.confirm.

Table 52 – DLME-CHANNEL-CONDITION.confirm parameters

Parameter	Data type	Valid range	Description

Status	Unsigned8	0 to 255	Result of the channel condition report request:
			0 = SUCCESS;
			1 = FAILURE;
			others are reserved.

8.3.6.4 Channel condition report process

The process of channel condition report is shown in Figure 42.

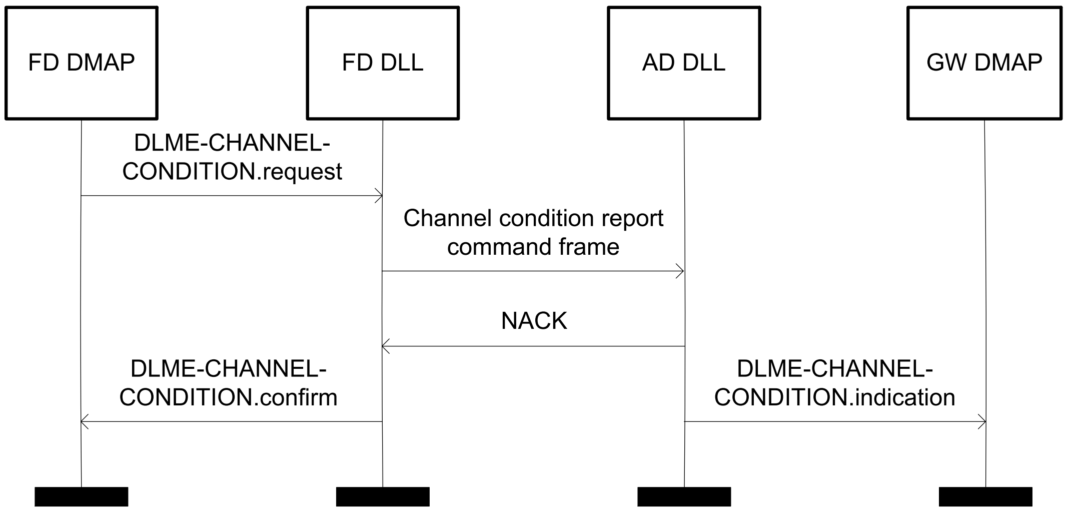
FD DMAP
FD DLL
AD DLL
GW DMAP
DLME-CHANNEL-CONDITION.request
Channel condition report command frame
NACK
DLME-CHANNEL-CONDITION.confirm
DLME-CHANNEL-CONDITION.indication

Figure 42 – Channel condition report process

The field device DMAP invokes DLME-CHANNEL-CONDITION.request in DLL, indicating DLL send channel condition report command frame; after receiving channel condition reports command frame, the access returns NACK, and reports to the gateway device DMAP by DLME-CHANNEL-CONDITION.indication; the field device DLL confirms to DMAP by DLME-CHANNEL-CONDITION.confirm after receiving NACK.

8.3.7 Remote attribute get services

8.3.7.1 DLME-INFO-GET.request

DLME-INFO-GET.request is used to remotely request attributes in the MIB.

```
DLME-INFO-GET.request(
                Handle,
                DstAddr,
                AttributeID,
                MemberID,
                FirstStoreIndex,
                Count
                )
```

Table 53 specifies the parameters for DLME-INFO-GET.request.

Table 53 – DLME-INFO-GET.request parameters

Parameter	Data type	Valid range	Description
Handle	Unsigned8	0 to 255	Assigned handle when invoking the DLME-INFO-GET.request
DstAddr	Unsigned16	0 to 65 535	The 8/16-bit short address of destination device
AttributeID	Unsigned8	0 to 255	AttributeID in the MIB
MemberID	Unsigned8	0 to 255	The identifier of attribute member, which is used to get the structured MIB attributes. The value 255 means that all attribute members should be read. This value is not valid for unstructured attributes
FirstStoreIndex	Unsigned16	0 to 65 535	The first storage index of multiple records, this value is not valid for the unstructured attributes
Count	Unsigned16	0 to 65 535	Number of records, which is used to get the MIB records, getting all records from FirstStoreIndex if Count = 0

8.3.7.2 DLME-INFO-GET.indication

DLME-INFO-GET.indication is used to inform the DMAP that attributes getting request frame has been successfully received.

```
DLME-INFO-GET.indication(
                          SrcAddr,
                          AttributeID,
                          MemberID,
                          FirstStoreIndex,
                          Count
                          )
```

Table 54 specifies the parameters for DLME-INFO-GET.indication.

Table 54 – DLME-INFO-GET.indication parameters

Parameter	Data type	Valid range	Description
SrcAddr	Unsigned16	0 to 65 535	The 8/16-bit short address of source device
AttributeID	Unsigned8	0 to 255	AttributeID in the MIB
MemberID	Unsigned8	0 to 255	The identifier of attribute member, which is used to get the structured MIB attributes. The value 255 means that all attribute members should be read. This value is not valid for the unstructured attributes
FirstStoreIndex	Unsigned16	0 to 65 535	The first storage index of multiple records, this value is not valid for the unstructured attributes
Count	Unsigned16	0 to 65 535	Number of records, which is used to get the MIB records, getting all records from FirstStoreIndex if Count = 0

8.3.7.3 DLME-INFO-GET.response

DLME-INFO-GET.response is used to respond to DLME-INFO-GET.indication.

```
DLME-INFO-GET.response(
                          DstAddr,
                          Status,
```

AttributeID,
MemberID,
FirstStoreIndex,
Count,
AttributeValue
)

Table 55 specifies the parameters for DLME-INFO-GET.response.

Table 55 – DLME-INFO-GET.response parameters

Parameter	Data type	Valid range	Description
DstAddr	Unsigned16	0 to 65 535	The 8/16-bit short address of destination device

Status	Unsigned8	0 to 255	Execution result of the request
			0 = SUCCESS;
			1 = UNSUPPORTED_ATTRIBUTE;
			Others are reserved.

AttributeID	Unsigned8	0 to 255	AttributeID in the MIB
MemberID	Unsigned8	0 to 255	The identifier of attribute member, which is used to get the structured MIB attributes. The value 255 means that all attribute members should be read. This value is not valid for the unstructured attributes
FirstStoreIndex	Unsigned16	0 to 65 535	The first storage index of multiple records, this value is not valid for the unstructured attributes
Count	Unsigned16	0 to 65 535	Number of records, which is used to get the MIB records, getting all records from FirstStoreIndex if Count = 0
AttributeValue	Octectstring		Value of the attribute to be read

If the operation of getting attributes is successful, the Status should be SUCCESS; if the MIB does not have the needed attributes, the Status should be UNSUPPORTED_ATTRIBUTE, and AttibuteValue is invalid.

8.3.7.4 DLME-INFO-GET.confirm

DLME-INFO-GET.confirm is used to return the results of DLME-INFO-GET.request.

DLME-INFO-GET.confirm(
Handle,
SrcAddr,
Status,
AttributeID,
MemberID,
FirstStoreIndex,
Count,
AttributeValue
)

Table 56 specifies the parameters for DLME-INFO-GET.confirm.

Table 56 – DLME-INFO-GET.confirm parameters

Parameter	Data type	Valid range	Description
Handle	Unsigned8	0 to 255	Assigned handle when invoking the DLME-INFO-GET.request
SrcAddr	Unsigned16	0 to 65 535	The 8/16-bit short address of source device

Status	Unsigned8	0 to 255	Execution result of the request
			0 = SUCCESS;
			1 = UNSUPPORTED_ATTRIBUTE;
			Others are reserved.

AttributeID	Unsigned8	0 to 255	AttributeID in the MIB
MemberID	Unsigned8	0 to 255	The identifier of attribute member, which is used to get the structured MIB attributes. The value 255 means that all attribute members should be read. This value is not valid for the unstructured attributes
FirstStoreIndex	Unsigned16	0 to 65 535	The first storage index of multiple records, this value is not valid for the unstructured attributes
Count	Unsigned16	0 to 65 535	Number of records, which is used to get the MIB records, getting all records from FirstStoreIndex if Count = 0
AttributeValue	Octectstring		Value of the attribute to be read

8.3.7.5 Remote attribute get process

The process of the remote attribute get process is shown in Figure 43.

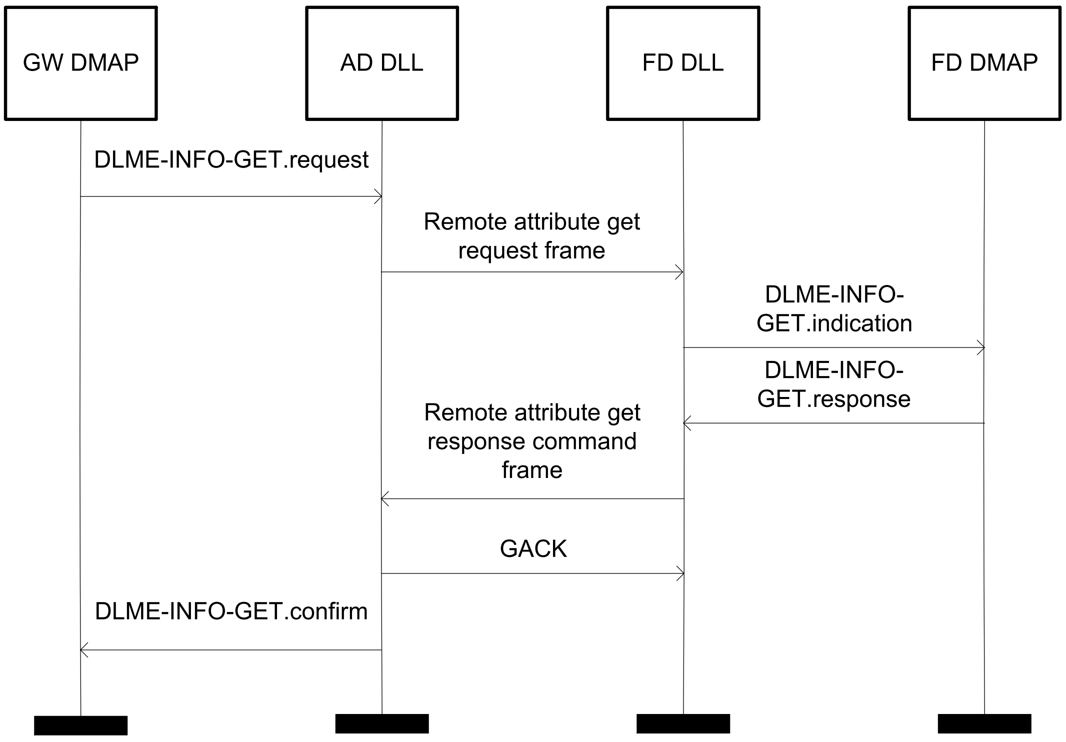

Figure 43 – Remote attribute get process

The gateway device DMAP invokes DLME-INFO-GET.request primitive in DLL, indicating the access device DLL sends remote attribute get request command frame; After receiving remote attribute get request command frame, the field device reports to DMAP by DLME-INFO-GET.indication; The field device DMAP invokes DLME-INFO-GET.response, indicating DLL send remote attribute get response command frame; After receiving remote attribute get response command frame, the access device returns GACK, and reports to DMAP by DLME-INFO-GET.confirm.

8.3.8 Remote attribute configuration services

8.3.8.1 DLME-INFO-SET.request

DLME-INFO-SET.request is used to remotely modify the MIB attribute values of field devices.

```
DLME-INFO-SET.request(
                        Handle,
                        DstAddr,
                        AttributeOption,
                        AttributeID,
                        MemberID,
                        FirstStoreIndex,
                        Count,
                        AttributeValue
                        )
```

Table 57 specifies the parameters for DLME-INFO-SET.request.

Table 57 – DLME-INFO-SET.request parameters

Parameter	Data type	Valid range	Description
Handle	Unsigned8	0 to 255	Assigned handle when invoking the DLME-INFO-SET.request
DstAddr	Unsigned16	0 to 65 535	The 8/16-bit short address of destination device

AttributeOption	Unsigned8	0 to 255	The operation of remote set attribute:
			0 = Add;
			1 = Delete;
			2 = Update.

AttributeID	Unsigned8	0 to 255	AttributeID in the MIB
MemberID	Unsigned8	0 to 255	The identifier of attribute member, which is used to set the structured MIB attributes. The value 255 means that all attribute members should be set. This value is not valid for the unstructured attributes
FirstStoreIndex	Unsigned16	0 to 65 535	The first storage index of multiple records, this value is not valid for the unstructured attributes
Count	Unsigned16	0 to 65 535	Number of records, which is used to set the MIB records, setting all records from FirstStoreIndex if Count = 0
AttributeValue	Octetstring		Value of the attribute to be written. If AttributeOption=1, this value is NULL

8.3.8.2 DLME-INFO-SET.indication

NLME-INFO-SET.indication is used to inform the DMAP of the successful receipt of an attribute setting request command packet.

```
DLME-INFO-SET.indication(
                        SrcAddr,
                        AttributeOption,
                        AttributeID,
                        MemberID,
                        FirstStoreIndex,
                        Count,
                        AttributeValue
                        )
```

Table 58 specifies the parameters for DLME-INFO-SET.indication.

Table 58 – DLME-INFO-SET.Indication parameters

Parameter	Data type	Valid range	Description
SrcAddr	Unsigned16	0 to 65 535	The 8/16-bit short address of source device

AttributeOption	Unsigned8	0 to 255	The operation of remote set attribute:
			0 = Add;
			1 = Delete;
			2 = Update.

AttributeID	Unsigned8	0 to 255	AttributeID in the MIB
MemberID	Unsigned8	0 to 255	The identifier of attribute member, which is used to set the structured MIB attributes. The value 255 means that all attribute members should be set. This value is invalid for the unstructured attributes
FirstStoreIndex	Unsigned16	0 to 65 535	The first storage index of multiple records, this value is not valid for the unstructured attributes
Count	Unsigned16	0 to 65 535	Number of records, which is used to set the MIB records, setting all records from FirstStoreIndex if Count = 0
AttributeValue	Octetstring		Value of the attribute to be written. If AttributeOption=1,this value is NULL

8.3.8.3 DLME-INFO-SET.response

DLME-INFO-SET.response is used to respond to DLME-INFO-SET.indication.

```
DLME-INFO-SET.response(
                    SrcAddr,
                    AttributeOption,
                    AttributeID,
                    MemberID,
                    FirstStoreIndex,
                    Count,
                    Status
                    )
```

Table 59 specifies the parameters for DLME-INFO-SET.response.

Table 59 – DLME-INFO-SET response parameters

Parameter	Data type	Valid range	Description
SrcAddr	Unsigned16	0 to 65 535	The 8/16-bit short address of source device

AttributeOption	Unsigned8	0 to 255	The operation of remote set attribute:
			0 = Add;
			1 = Delete;
			2 = Update.

AttributeID	Unsigned8	0 to 255	AttributeID in the MIB
MemberID	Unsigned8	0 to 255	The identifier of attribute member, which is used to set the structured MIB attributes. The value 255 means that all attribute members should be set. This value is invalid for the unstructured attributes
FirstStoreIndex	Unsigned16	0 to 65 535	The first storage index of multiple records, this value is not valid for the unstructured attributes
Count	Unsigned16	0 to 65 535	Number of records, which is used to set the MIB records, setting all records from FirstStoreIndex if Count = 0

Status	Unsigned8	0 to 255	Result of remote set attribute:
			0 = SUCCESS;
			1 = UNSUPPORTED_ATTRIBUTE;
			2 = INVALID_PARAMETER;
			Others are reserved.

If the operation of remote attributes set is successful, the Status should be SUCCESS; if the MIB does not have the needed attributes, the Status should be UNSUPPORTED_ATTRIBUTE; if the number of records does not equal to the required count, the Status should be INVALID_PARAMETER.

8.3.8.4 DLME-INFO-SET.confirm

DLME-INFO-SET.confirm is used to return the results of DLME-INFO-SET.request.

```
DLME-INFO-SET.confirm(
                        Handle,
                        Status
                        )
```

Table 60 specifies the parameters for DLME-INFO-SET.confirm.

Table 60 – DLME-INFO-SET.confirm parameters

Parameter	Data type	Valid range	Description
Handle	Unsigned8	0 to 255	Assigned handle when invoking the DLME-INFO-SET.request

Status	Unsigned8	0 to 255	Result of remote set attribute:
			0 = SUCCESS;
			1 = UNSUPPORTED_ATTRIBUTE;
			2 = INVALID_PARAMETER;
			Others are reserved.

8.3.8.5 Remote attribute Configuration process

The process of the remote attribute set is shown in Figure 44.

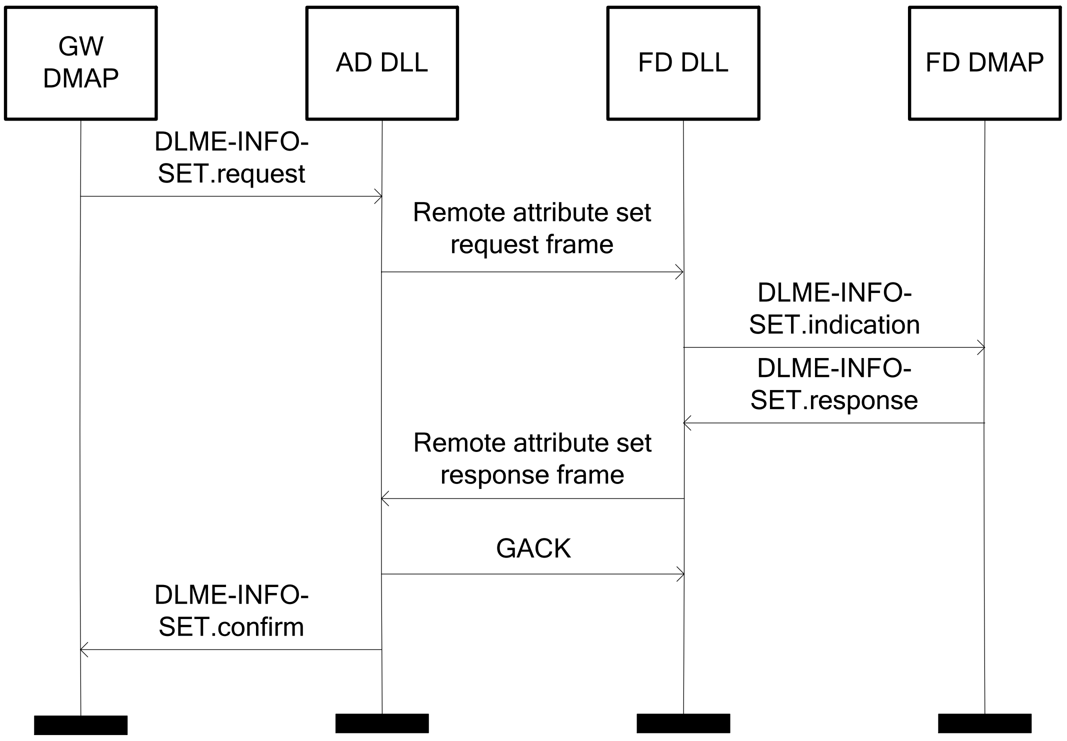

Figure 44 – Remote attribute set process

The gateway device DMAP invokes DLME-INFO-SET.request primitive in DLL, indicating the access device DLL sends remote attribute set request command frame; After receiving remote attribute set request command frame, the field device reports to DMAP by DLME-INFO-SET.indication; The field device DMAP invokes DLME-INFO-SET.response, indicating DLL send remote attribute set response command frame; after receiving remote attribute set response command frame, the access device returns GACK, and reports to DMAP by DLME-INFO-SET.confirm.

8.3.9 Device leaving services

8.3.9.1 DLME-LEAVE.request

DLME-LEAVE.request is used for a field device to leave WIA-FA network.

```
DLME-LEAVE.request(
                   ShortAddr
                   )
```

Table 61 specifies the parameters for DLME-LEAVE.request.

Table 61 – DLME-LEAVE.request parameters

Parameter	Data type	Valid range	Description
ShortAddr	Unsigned16	0 to 255/0 to 65 535	The short address of the device asked to leave, (see DeviceShortAddress in 6.7.1.2.1)

8.3.9.2 DLME-LEAVE.indication

DLME-LEAVE.indication is used to notify the field device that it has received a device leave request.

DLME-LEAVE.indication(
)

8.3.9.3 DLME-LEAVE.confirm

DLME-LEAVE.confirm is used to report the result of DLME-LEAVE.request.

DLME-LEAVE.confirm(
Status
)

Table 62 specifies the parameters for DLME-LEAVE.confirm.

Table 62 – DLME-LEAVE.confirm parameters

Parameter	Data type	Valid range	Description

Status	Unsigned8	0 to 255	Result of leave request:
			0 = SUCCESS;
			1 = FAILURE;
			Others are reserved.

8.3.9.4 Device leave process

Device leave process is shown in Figure 45.

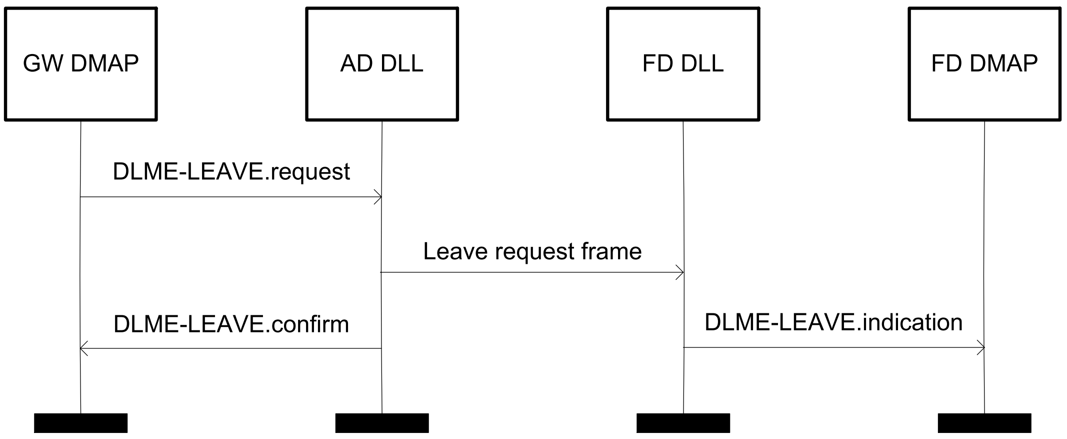

Figure 45 – Device leave process

The gateway device DMAP invokes DLME-LEAVE.request primitive in DLL, indicating the access device DLL to send leave request command frame; After receiving leave request command frame, the field device reports to DMAP by DLME-LEAVE.indication. The access device DLL reports sending leave request command frame successfully to the gateway device by DLME-LEAVE.confirm.

8.4 DLL frame formats

8.4.1 General frame format

The DLL general frame format is illustrated in Figure 46.

DLPDU		FCS
DLL frame header	DLL payload	

Figure 46 – General frame format

The DLL frame is composed of:

– WIA-FA DLL frame header, see Figure 47;

– DLL payload;

– Frame Check Sequence (FCS).

NOTE See Clause 11 for Security.

1 octet	1 octet	1/ 2/8 octets	2 octets	2 octets
Frame control	Network ID	Peer address	Sequence number	Frame length

Figure 47 – DLL frame header

The DLL frame header field has the following subfields:

– Frame Control, see Figure 48;

– Network ID: this field has 1-octet length, distinguish multiple network;

– Peer address: this field has 1 or 2 or 8-octet length, 1 or 2-octet length indicate short address, 8-octet length indicate long address;

– Sequence number: this field has 2-octet length, this field specifies frame sequence, increase from 1 to maximum and reset;

– Frame length: this field has 2-octet length, this field specifies the DLL payload length;

WIA-FA DLL frame control format is shown in Figure 48.

Bit: 0 to 4	Bit: 5	Bit: 6	Bit: 7
Frame type	Reserved	Preemption flag	Address mode

Figure 48 – DLL frame control format

– Frame Type field has 4-bit length, defined in Table 63;

Table 63 – Frame type coding

Bit: 0 to 4	Frame Type
00000	Beacon frame
00001	Data frame
00010	Aggregation frame
00011	GACK
00100	NACK
00101	Join request frame
00110	Join response frame
00111	Leave request frame
01000	Device status report frame
01001	Channel condition report frame
01010	Time synchronization request frame
01011	Time synchronization response frame
01100	Remote attribute get request frame
01101	Remote attribute get response frame
01110	Remote attribute set request frame
01111	Remote attribute set response frame
10000	Key establish request frame (see 11.7.3)
10001	Key establish response frame (see 11.7.4)
10010	Key update request frame (see 11.7.5)
10011	Key update response frame (see 11.7.6)
10100	Security alarm request frame (see 11.7.7)
10101 to 11111	Reserved

– Preemption Flag subfield has 1-bit length, this field specifies whether the frame is a occupied frame or not;

– Address Mode subfield has 1-bit length, this field specifies the type of source address and destination address in the DLL frame header, see Table 64.

Table 64 – Addressing mode subfields

Bit: 7	Description
0	64-bit long address
1	8/16-bit short address

8.4.2 Date frame format

The format of the data frame is shown in Figure 49.

7/8 octets	Variable Length
DLL frame header	Data frame payload

Figure 49 – DLL Date frame format

– DLL frame header: see Figure 47;

– Data frame payload: this field has variable length, this field specifies the content of data frame, see Payload in 8.4.2.

8.4.3 Aggregation frame format

The format of the DLL Aggregation frame is shown in Figure 50.

7/8 octets	Variable Length
DLL frame header	Aggregation frame payload

Figure 50 – DLL Aggregation frame format

– DLL frame header: see Figure 47;

– Aggregation frame payload: this field has variable length, this field specifies the content of Aggregation frame, see Figure 30 in 8.1.5.

8.4.4 NACK frame format

The format of the NACK frame format is shown in Figure 51.

7/8 octets	1 octet	Variable Length
DLL frame header	Retransmitting devices count	Short address list

Figure 51 – NACK frame format

– DLL frame header: see Figure 47;

– Retransmitting devices count: this field has 1-octet length, indicating the count of field devices required retransmitting;

– Short address list: this field has variable length, indicating the list of short addresses that field devices required retransmitting.

8.4.5 GACK frame format

The format of the GACK frame format is shown in Figure 52.

7/8/14 octets	1 octet	Variable Length
DLL frame header	Device count	GACK information

Figure 52 – GACK frame format

1/2octets	2 octets
Devices short address	Sequence number

Figure 53 – GACK information

– DLL frame header: see Figure 47;

– Device count: this field is 1 octet length, indicating the count of devices transmitting frames to any access device;

– GACK information: this field is variable, which includes Device short address and Sequence number in Figure 53.

– Sequence number: this field has 2-octet length, indicating the sequence number of a frame received from a field device.

8.4.6 Beacon frame format

The general format of the DLL beacon frame is shown in Figure 54.

7/8 octets	2 octets	2 octets	2 octets	2 octets	1 octet	8 octets	Variable Length
DLL frame header	Super frame length	Timeslot duration	Beacon frame relative timeslot number	First shared timeslot number	Shared timeslot count (see Figure 55)	Absolute time value	Beacon frame payload

Figure 54 – DLL Beacon frame format

Bit: 0 to 3	Bit: 4 to 7
Uplink shared timeslot number	Downlink timeslot number

Figure 55 – Shared timeslot count

- DLL frame header: see Figure 47;
- Super frame length: this field has 2-octet length, this field specifies the length of the default superframe, see 8.1.2;
- Timeslot duration: this field has 2-octet length, this field specifies the timeslot length that has be configured, see TimeSlotDuration in 6.7.1.2.1;
- Beacon frame relative timeslot number: this field has 2-octet length, this field specifies the timeslot sending beacon, see 8.1.2;
- First shared timeslot number: this field has 2-octet length, this field specifies the shared timeslot in superframe, see 8.1.2;
- Shared timeslot number: this field has 1-octet length, this field specifies the total number of shared timeslot, see Figure 55; 0 to 3 bit specifies the uplink slot number, the field device sends join request to gateway device, and sends remote attribute get/set response in these timeslots; 4 to 7 bit specifies the count of the downlink timeslots that are used by the gateway device to send the join response, the gateway device sends join response, remote attribute get/set request, and application configuration in these timeslots, see 8.1.2;
- Absolute time value: this field has 8-octet length, and specifies the absolute time sending beacon; see TimeValue in 6.7.1.2.1 for the value of this field;
- Beacon frame payload: this field has variable length, this field specifies the payload in beacon frame.

8.4.7 Join request frame format

The general format of the DLL join request frame is shown in Figure 56.

14 octets	0/8 octets
DLL frame header	Security Material

Figure 56 – DLL join request frame format

- DLL frame header: see Figure 47;
- Security Material: this field has 8-octet length, this field specifies the authentication information of the new device waiting to join; see SecMaterial in 8.3.4.1 for the value of this field.

8.4.8 Join response frame format

The general format of the DLL join response frame is shown in Figure 57.

14 octets	1 octet	1/2 octets
DLL frame header	Status	Allocated short address

Figure 57 – DLL join request frame format

- DLL frame header: see Figure 47;
- Status: this field has 1-octet length, this field specifies the joining status of field device; see Status in 8.3.4.3 for the value of this field;
- Allocated short address: this field has 1 or 2-octet length and specifies short address allocated by the GW to device waiting to join, see ShortAddr in 8.3.4.3 for the value of this field.

8.4.9 Leave request frame format

The general format of the DLL leave request frame is shown in Figure 58.

7/8 octets
DLL frame header

Figure 58 – DLL leave request frame format

– DLL frame header: see Figure 47.

8.4.10 Device status report frame format

The general format of the DLL Device status report frame is shown in Figure 59.

7/8 octets	Variable Length
DLL frame header	Device status

Figure 59 – DLL Device status report frame format

– DLL frame header: see Figure 47;

– Device status: this field has Variable length and specifies the status of device; see DeviceConditionInfo in 8.3.5.1 for the value of this field.

8.4.11 Channel condition report frame format

The general format of the DLL Channel condition report frame is shown in Figure 60.

7/8 octets	Variable Length
DLL frame header	Channel condition

Figure 60 – DLL Channel condition report frame format

– DLL frame header: see Figure 47;

– Channel condition: this field has Variable length and specifies the condition of channel; see ChannelConditionInfo in 8.3.6.1 for the value of this field.

8.4.12 Time synchronization request frame format

The format of the DLL time synchronization request frame is shown in Figure 61.

7/8 octets	8 octets
DLL frame header	Field Device Time Value

Figure 61 – DLL time synchronization request frame format

– DLL frame header: see Figure 47;

– Field Device Time Value: this field has 8-octet length, this field specifies the timestamp when the field device sends time synchronization request command frame; see FieldDeviceTimeValue in 8.3.3.3 for the value of this field.

8.4.13 Time synchronization response frame format

The format of the DLL time synchronization response frame is shown in Figure 62.

7/8 octets	8 octets	8 octets
DLL frame header	Field Device Time Value	Receive Time Value

Figure 62 – DLL time synchronization response frame format

– DLL frame header: see Figure 47;

– Field Device Time Value and its value: see 8.4.12;

– Receive Time Value: this field has 8-octet length; this field specifies the timestamp when the access device receives the time synchronization request frame; see ReceiveTimeValue in 8.3.3.3 for the value of this field.

8.4.14 Remote attribute get request frame format

The format of the DLL remote attribute get request frame is shown in Figure 63.

7/8 octets	1 octets	1 octets	2 octets	2 octets
DLL frame header	AttributeID	Attribute member ID	First storage index of multiple attribute values	Number of attributes

Figure 63 – DLL Remote attribute get request frame format

– DLL frame header: see Figure 47;

– AttributeID: this field has 1-octet length; see AttributeID in 8.3.7.1 for the value of this field;

– Attribute member ID: this field has 1-octet length; see MemberID in 8.3.7.1 for the value of this field;

– First storage index of multiple attribute values: this field has 2-octet length; see FirstStoreIndex in 8.3.7.1 for the value of this field;

– Number of attributes: this field has 2-octet length; see Count in 8.3.7.1 for the value of this field.

8.4.15 Remote attribute get response frame format

The format of the DLL remote attribute get response frame is shown in Figure 64.

7/8 octets	1 octets	1 octets	1 octets	2 octets	2octets	Variable Length
DLL frame header	Status	AttributeID	Attribute member ID	First storage index of multiple attribute values	Number of attributes	Attribute Value

Figure 64 – DLL remote attribute get response frame format

– DLL frame header: see Figure 47;

– Status: this field has 1-octet length; see Status in 8.3.7.3 for the value of this field;

– AttributeID: this field has 1-octet length; see AttributeID in 8.3.7.3 for the value of this field;

– Attribute member ID: this field has 1-octet length; see MemberID in 8.3.7.3 for the value of this field;

– First storage index of multiple attribute values: this field has 2-octet length; see FirstStoreIndex in 8.3.7.3 for the value of this field;

– Number of attributes: this field has 2-octet length; see Count in 8.3.7.3 for the value of this field;

– Attribute Value: this field has variable length; see AttributeValue in 8.3.7.3 for the value of this field.

8.4.16 Remote attribute set request frame format

The format of the DLL remote attribute set request frame is shown in Figure 65.

7/8 octets	1 octets	1 octets	1 octets	2 octets	2octets	Variable Length
DLL frame header	AttributeOption	AttributeID	Attribute member ID	First storage index of multiple attribute values	Number of attributes	Attribute Value

Figure 65 – DLL Remote attribute set request frame format

– DLL frame header: see Figure 47;

– AttributeOption: this field has 1-octet length; see AttributeOption in 8.3.8.1 for the value of this field;

– AttributeID: this field has 1-octet length; see AttributeID in 8.3.8.1 for the value of this field;

– Attribute member ID: this field has 1-octet length; see MemberID in 8.3.8.1 for the value of this field;

– First storage index of multiple attribute values: this field has 2-octet length; see FirstStoreIndex in 8.3.8.1 for the value of this field;

– Number of attributes: this field has 2-octet length; see Count in 8.3.8.1 for the value of this field;

– AttributeValue: this field has variable length; see AttributeValue in 8.3.8.1 for the value of this field.

8.4.17 Remote attribute set response frame format

The format of the DLL remote attribute set response frame is shown in Figure 66.

7/8 octets	1 octets	1 octets	1 octets	2 octets	2octets	1 octets
DLL frame header	AttributeOption	AttributeID	Attribute member ID	First storage index of multiple attribute values	Number of attributes	Status

Figure 66 – DLL remote attribute set response frame format

– DLL frame header: see Figure 47;

– AttributeOption: this field has 1-octet length; see AttributeOption in 8.3.8.3 for the value of this field;

– AttributeID: this field has 1-octet length; see AttributeID in 8.3.8.3 for the value of this field;

– Attribute member ID: this field has 1-octet length; see MemberID in 8.3.8.3 for the value of this field;

– First storage index of multiple attribute values: this field has 2-octet length; see FirstStoreIndex in 8.3.8.3 for the value of this field;

– Number of attributes: this field has 2-octet length; see Count in 8.3.8.3 for the value of this field;

– Status: this field has 1-octet length; see Status in 8.3.8.3 for the value of this field.

8.5 Data link layer state machines

8.5.1 DLL state machine of access field

The state machine of an access field is shown in Figure 67.

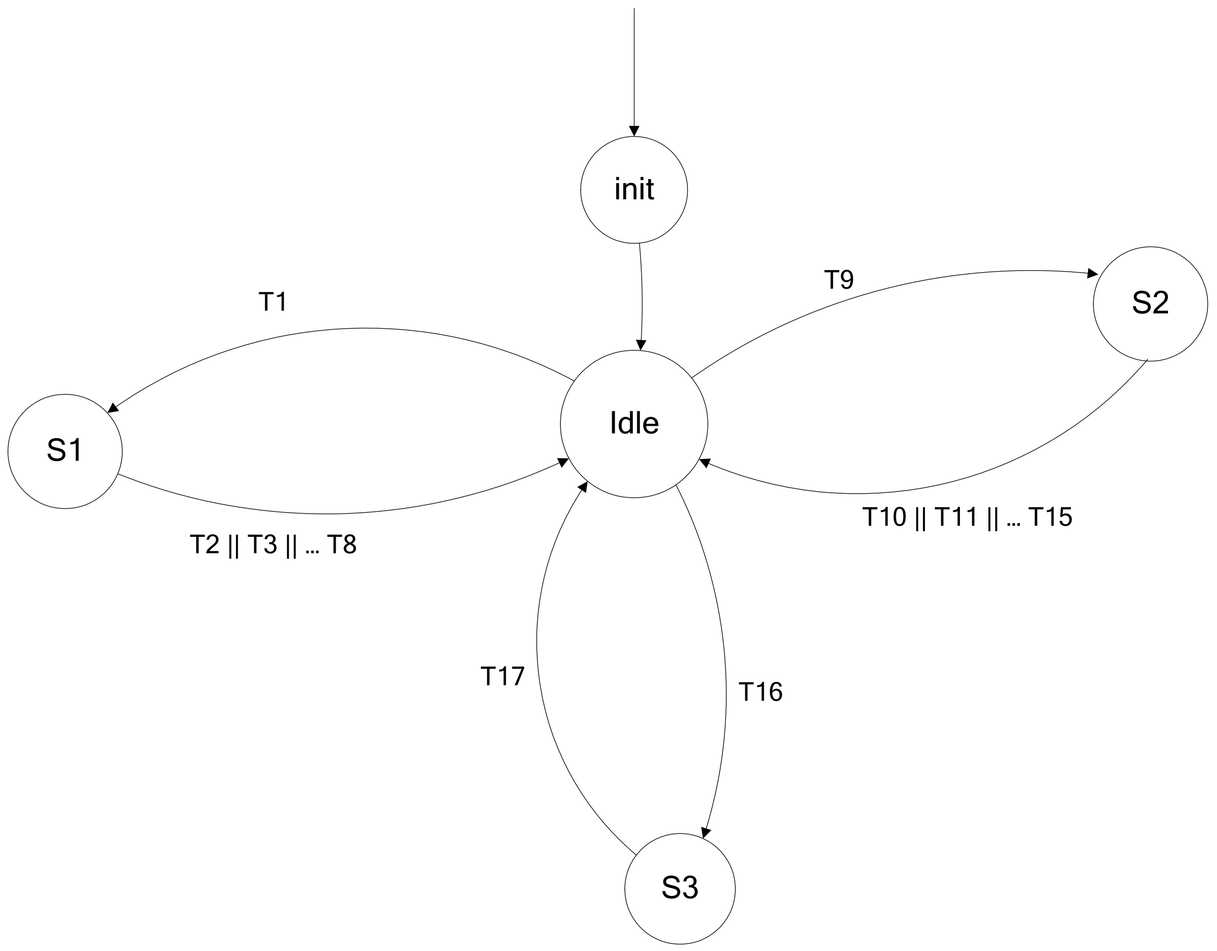

Figure 67 – DLL state machine of access device

The DLL state transition of an access device is listed in Table 65.

Table 65 – DLL state transition of access device

#	Current State	Event or condition => \action	Next state
T1	Idle	(PrimitiveType == PHY-DATA.indication) => FrameType = GetFrameType(DLPDU);	S1

T2	S1	(FrameType == JoinRequest Command)
		=>
		PhyAddr = GetPhyAddr(DLPDU);
		DLME-JOIN.indication(PhyAddr, SecMaterial);
			Idle

T3	S1	(FrameType == Data)
		=>
		SrcAddr = GetSrcAddr(DLPDU);
		PayloadLength = GetPayloadLength(DLPDU);
		Payload = GetPayload(DLPDU);
		DLDE-DATA.indication(SrcAddr, DataType:= Data, PayloadLength, Payload);
			Idle

T4	S1	(FrameType == RemoteAttributeSetResponse) => DLME-INFO-SET.confirm(Handle, Status);	Idle
T5	S1	(FrameType == RemoteAttributeGetResponse) => DLME-INFO-GET.confirm(Handle, Status);	Idle
T6	S1	(FrameType == DeviceStatusReport) => DLME-DEVICE-STATUS.indication(PowerSupplyStatus)	Idle
T7	S1	(FrameType == ChannelConditionReport) => DLME-CHANNEL-STATUS.indication(ShortAddr, ChannelConditionInfo);	Idle
T8	S1	(FrameType == TowWayTimeSynchronizationRequest) => DLME-TIME-SYN.indication(ShortAddr, FieldDeviceTimeValue);	Idle
T9	Idle	APP or DMAP invokes primitives of DLL	S2
T10	S2	(PrimitiveType == DLME-JOIN.response) => BuildFrame(FrameType:= JoinResponse);	Idle
T11	S2	(PrimitiveType == DLME-INFO-SET.request) => BuildFrame(FrameType:= RemoteAttributeSetRequest);	Idle
T12	S2	(PrimitiveType == DLME-INFO-GET.request) => BuildFrame(FrameType:= RemoteAttributeGetRequest);	Idle
T13	S2	(PrimitiveType == DLDE-DATA.request) => BuildFrame(FrameType:= Data);	Idle

#	Current State	Event or condition => \action	Next state
T14	S2	(PrimitiveType == DLME-TIME-SYN.response) => BuildFrame(FrameType:= TowWayTimeSynchronizationResponse);	Idle
T15	S2	(PrimitiveType == DLME-LEAVE.request) => BuildFrame(FrameType:= LeaveRequest);	Idle

T16	Idle	Slot timeout
		=>
		If (LinkType == TRANSMIT_LINK)
		{
		Phy_set_RF_mode(TRANSMIT_MODE);
		DLPDU = GetDLPDUFromQueue();
		PHY-DATA.request(DLPDU);
		}
		Else if (LinkType == RECEIVE_LINK)
		{
		  Phy_set_RF_mode(RECEIVE_MODE);
		}
		Else if(LinkType == SHARED_TRANSMIT_LINK)
		{
		   PHYSendWithBackoff(DLPDU);
		}
			S3

T17	S3	(TransmissionCompleteISR()) || (ReceiveCompleteISR()) =>	Idle

– Init state

In Init state, The DLL of an access device initializes itself and enters to Idle state.

– Idle state

In Idle state, the following events shall occur:

a) PHY invokes PHY-DATA.indication to pass the received frame to DLL. DLL unpacks the frame and obtains DLPDU. The state machine enters to S1 state.

b) APP or DMAP invokes the DLL primitives. The state machine enters to S2 state.

If timeslot is timeout, DLL sets RF into transmission or receive mode according to the corresponding link type (see LinkType in Table 17) and performs transmission or reception. The state machine enters to S3 state.

– S1 state

In S1 state, DLL parses the frame type (see 8.4.1) of an unpacked frame, and triggers different conditions and executes different actions according to the frame type of DLPDU. Finally, the state machine enters to Idle state.

a) DLL invokes DLME-JOIN.indication after receiving a join request frame (see 8.4.7). The state machine enters to Idle state.

b) DLL invokes DLDE-DATA.indication after receiving a data frame (see 8.4.2). The state machine enters to Idle state.

c) DLL invokes DLME-INFO-SET.confirm after receiving a remote attribute set response frame (see 8.4.17). The state machine enters to Idle state.

d) DLL invokes DLME-INFO-GET.confirm after receiving a remote attribute get response frame (see 8.4.15). The state machine enters to Idle state.

e) DLL invokes DLME-TIME-SYN.indication after receiving a two-way time synchronization request frame. The state machine enters to Idle state.

f) DLL invokes DLME-DEVICE-STATUS.indication after receiving a device status report frame (see 8.4.10). The state machine enters to Idle state.

g) DLL invokes DLME-CHANNEL-STATUS.indication after receiving a channel condition report frame (see 8.4.11). The state machine enters to Idle state.

– S2 state

In S2 state, the following events shall occur:

a) DMAP invokes DLME-JOIN.response to instruct DLL to generate a join response frame (see 8.4.8) and post it to the transmission queue. The state machine enters to Idle state.

b) DMAP invokes DLME-INFO-SET.request to request DLL generating a remote attribute set request frame (see 8.4.16) and post it to the transmission queue. The state machine enters to Idle state.

c) DMAP invokes DLME-INFO-GET.request to request DLL generating a remote attribute get request frame (see 8.4.14) and post it to the transmission queue. The state machine enters to Idle state.

d) DMAP invokes DLDE-DATA.request to request DLL generating data frame (see 8.4.2) and post it to the transmission queue. The state machine enters to Idle state.

e) DMAP invokes DLME-LEAVE.request to request DLL generating a leave request frame (see 8.4.9) and post it to the transmission queue. The state machine enters to Idle state.

f) DMAP invokes DLME-TIME-SYN.response to instruct DLL to generate a two-way time synchronization response frame (see 8.4.13) and post it to the transmission queue. The state machine enters to Idle state.

– S3 state

In S3 state, DLL sets RF to transmission or reception mode according to link types (see LinkType in Table 17). If the link type is receiving, DLL sets the RF to reception mode; if link type is transmitting, DLL sets RF into transmission mode and invokes PHY-DATA.request to transmit a frame in transmission queue; if the link type is transmit-shared, DLL invokes PHYSendWithBackoff function to transmit it by using a backoff algorithm. After the interrupt handle of a transmission or reception is executed, the state machine enters to Idle state.

8.5.2 DLL state machine of field device

The state machine of a field device is shown in Figure 68.

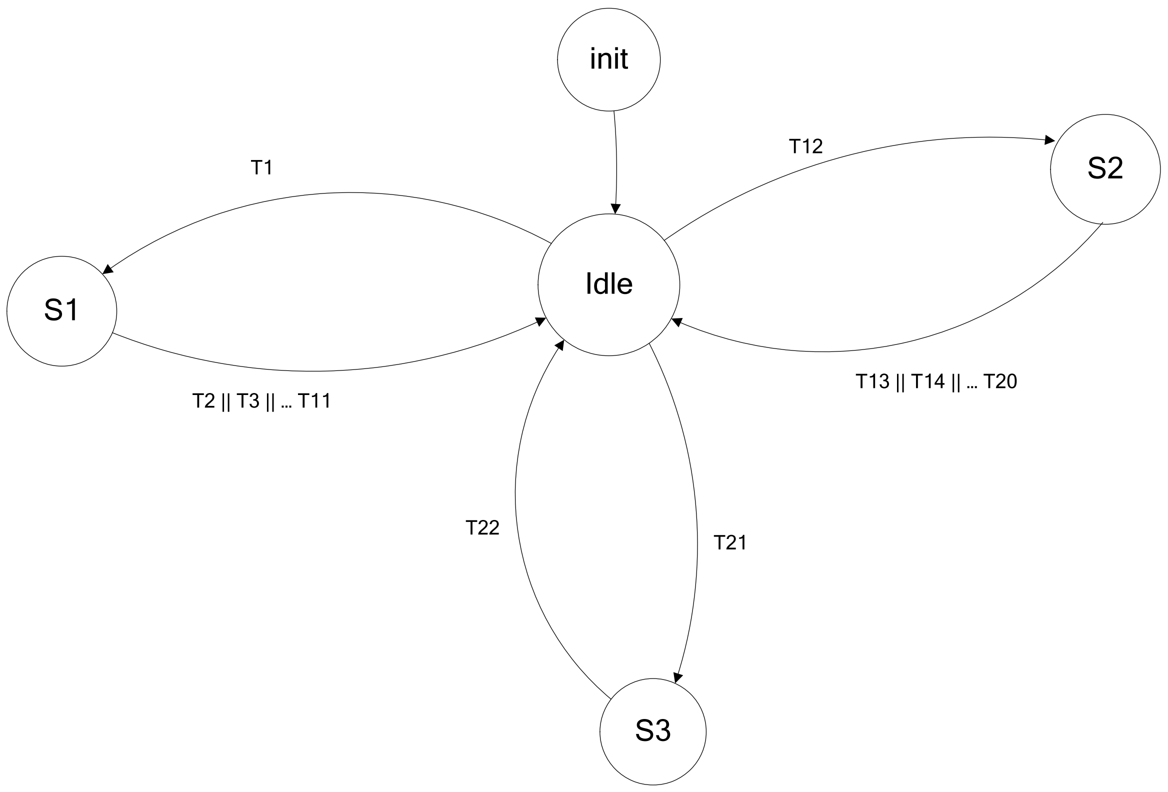

Figure 68 – DLL state machine of field device

The DLL state transition of a field device is listed in Table 66.

Table 66 – DLL state transition of field device

#	Current State	Event or condition => \action	Next state
T1	Idle	(PrimitiveType == PHY-DATA.indication) => FrameType = GetFrameType(DLPDU);	S1
T2	S1	(FrameType == JoinResponse) => DLME-JOIN.confirm(Status, ShortAddr);	Idle
T3	S1	(FrameType == RemoteAttributeSetRequest) => DLME-INFO-SET.indication(Handle, AttributeOption, AttributeID, AttributeMemID, FirstStoreIndex, Count, AttributeValue);	Idle
T4	S1	(FrameType == RemoteAttributeGetRequest) => DLME-INFO-GET.indication(Handle, Attribute,AttributeMemID, FirstStroreIndex, Count);	Idle
T5	S1	(FrameType ==Data) => DLDE-DATA.indication(SrcAddr, DataType, PayloadLength, Payload);	Idle
T6	S1	(FrameType == LeaveRequest) => DLME-LEAVE.indication();	Idle
T7	S1	(FrameType == NACK) && (NACK is for DeviceStatusReportCommand) => DLME-DEVICE-STATUS.confirm(Status);	Idle
T8	S1	(FrameType == NACK) && (NACK is for ChannelConditionReportCommand) => DLME-CHANNEL-STATUS.confirm(Status);	Idle

T9	S1	(FrameType == Beacon) && (device status == NOT_JOINED) =>
		DoSynchronization();
		DLME-DISCVOERY.confirm(status:= SUCCESS)
			Idle

T10	S1	(FrameType == Beacon) && (device status != NOT_JOINED) => DoSynchronization();	Idle
T11	S1	(FrameType == TowWayTimeSynchronizationResponse) => DLME-TIME-SYN.confirm(Status, FieldDeviceTimeValue, ReceiveTimeValue);	Idle
T12	Idle	APP or DMAP invokes primitives of DLL	S2
T13	S2	(PrimitiveType == DLME-JOIN.request) => BuildFrame(FrameType:= JoinRequest);	Idle
T14	S2	(PrimitiveType == DLME-INFO-SET.response) => BuildFrame(FrameType:= RemoteAttributeSetResponse);	Idle

#	Current State	Event or condition => \action	Next state
T15	S2	(PrimitiveType == DLME-INFO-GET.response) => BuildFrame(FrameType:= RemoteAttributeGetResponse);	Idle
T16	S2	(PrimitiveType == DLDE-DATA.request) => BuildFrame(FrameType:= Data);	Idle
T17	S2	(PrimitiveType == DLME-DISCOVERY.request) => ScanChanels(Channels);	Idle
T18	S2	(PrimitiveType == DLME-DEVICE-STATUS.request) => BuildFrame(FrameType:= DeviceStatusReport);	Idle
T19	S2	(PrimitiveType == DLME-TIME-SYN.request) => BuildFrame(FrameType:= TowWayTimeSynchronizationRequest);	Idle
T20	S2	(PrimitiveType == DLME-CHANNEL-STATUS.request) => BuildFrame(FrameType:= ChannelStatusReport);	Idle

T21	Idle	Slot timeout
		=>
		If (LinkType == TRANSMIT_LINK)
		{
		Phy_set_RF_mode(TRANSMIT_MODE);
		DLPDU = GetDLPDUFromQueue();
		PHY-DATA.request(DLPDU);
		}
		Else if (LinkType == RECEIVE_LINK)
		{
		  Phy_set_RF_mode(RECEIVE_MODE);
		}
		Else if(LinkType == SHARED_TRANSMIT_LINK)
		{
		PHYSendWithBackoff(DLPDU);
		}
			S3

T22	S3	(TransmissionCompleteISR()) || (ReceiveCompleteISR()) =>	Idle

– Init state

In Init state, The DLL of a field device initializes itself and enters to Idle state.

– Idle state

In Idle state, the following events shall occur:

a) PHY invokes PHY-DATA.indication to pass the received frame to DLL. DLL unpacks the frame and obtains DLPDU. The state machine enters to S1 state.

b) APP or DMAP invokes the DLL primitives. The state machine enters to S2 state.

c) If timeslot is timeout, DLL sets RF into transmission or receive mode according to the corresponding link type (see LinkType in Table 17) and performs transmission or reception. The state machine enters to S3 state.

– S1 state

In S1 state, DLL parses the frame type (see 8.4.1) of an unpacked frame, and triggers different conditions and executes different actions according to the frame type of DLPDU. Finally, the state machine enters to Idle state.

a) DLL invokes DLME-JOIN.confirm after receiving a join response frame (see 8.4.8). The state machine enters to Idle state.

b) DLL invokes DLDE-DATA.indication after receiving a data frame (see 8.4.2). The state machine enters to Idle state.

c) DLL invokes DLME-INFO-SET.indication after receiving a remote attribute set request frame (see 8.4.16). The state machine enters to Idle state.

d) DLL invokes DLME-INFO-GET.indication after receiving a remote attribute get request frame (see 8.4.14). The state machine enters to Idle state.

e) DLL invokes DLME-LEAVE.indication after receiving a leave request frame (see 8.4.9). The state machine enters to Idle state.

f) DLL invokes DLME-TIME-SYN.confirm after receiving a two-way time synchronization response frame (see 8.4.13). The state machine enters to Idle state.

g) DLL invokes DLME-DEVICE-STATUS.indication after receiving a NACK frame (see 8.4.4). The state machine enters to Idle state.

h) DLL invokes DLME-CHANNEL-STATUS.indication after receiving a NACK frame (see 8.4.4). The state machine enters to Idle state.

i) DLL makes time synchronization and invokes DLME-DISCOVERY.confirm after a field device attempt to join WIA-FA network receives a beacon frame (see 8.4.6). The state machine enters to Idle State.

– S2 state

In S2 state, the following events shall occur:

a) DMAP invokes DLME-JOIN.request to request DLL generating a join request frame (see 8.4.7) and post it to the transmission queue. The state machine enters to Idle state.

b) DMAP invokes DLME-INFO-SET.response to instruct DLL to generate a remote attribute set response frame (see 8.4.17) and post it to the transmission queue. The state machine enters to Idle state.

c) DMAP invokes DLME-INFO-GET.response to instruct DLL to generate a remote attribute get response frame (see 8.4.15) and post it to the transmission queue. The state machine enters to Idle state.

d) DMAP invokes DLDE-DATA.request to request DLL generating data frame (see 8.4.2) and post it to the transmission queue. The state machine enters to Idle state.

e) DMAP invokes DLME-DISCOVERY.request to request DLL setting RF to reception mode and scanning channels according to the parameters in DLME-DISCOVERY.request. The state machine enters to Idle state.

f) DLL invokes DLME-DEVICE-STATUS.request to request DLL generating a device status report frame (see 8.4.10). The state machine enters to Idle state.

g) DLL invokes DLME-CHANNEL-STATUS.request to request DLL generating a channel condition report frame (see 8.4.11). The state machine enters to Idle state.

– S3 state

In S3 state, DLL sets RF to transmission or reception mode according to link types (see LinkType in Table 17). If the link type is receiving, DLL sets the RF to reception mode; if link type is transmitting, DLL sets RF into transmission mode and invokes PHY-DATA.request to transmit a frame in transmission queue; if the link type is transmit-shared, DLL invokes PHYSendWithBackoff function to transmit it by using a backoff algorithm. After the interrupt handle of a transmission or reception is executed, the state machine enters to Idle state.

8.5.3 Functions used in DLL state transitions

The functions used in DLL state transtitions are listed in Table 67.

Table 67 – Functions used in DMAP state transition

Function	Input	Output	Description
GetFrameType()	DLPDU	JoinRequest Data RemoteAttributeSetResponse RemoteAttributeGetResponse DeviceStatusReport ChannelCondition Report JoinResponse RemoteAttributeSetRequest RemoteAttributeGetRequest Data LeaveRequest NACK GACK Beacon	Parsing frame type
GetPhyAddr()	DLPDU	PhyAddr	Parsing physical address
GetSrcAddr()	DLPDU	SrcAddr	Parsing source address
GetPayloadLength()	DLPDU	PayloadLength	Parsing payload length
BuildFrame()	FrameType		Generating frame and putting into queue
Phy_set_RF_mode()	mode		Setting RF mode
GetDLPDUFromQueue()		DLPDU	Taking DLPDU from transmission queue
TransmissionCompleteISR()			Interrupt function after transmission
ReceiveCompleteISR()			Interrupt function after receiving
Do_Backoff()		ShareSendSlot	Performing backoff algorithm
PHY-CCA()		BUSY IDLE	Performing PHY CCA
DoSynchronization()			Performing time synchronization
ScanChanels()	Channels		Scanning channels
PHYSendWithBackoff()	DLPDU		Sending data by using backoff

9 Wired specifications between GW and AD

9.1 Overview

The wired services and frame formats between the gateway device and access devices are defined in this PAS. The wired communication methods are beyond the scope of this PAS.

9.2 Join process of access device

An access device does not scan channels when it attempts to join a WIA-FA network.

The join process of an access device is performed as follows.

– The access device sends a join request command frame to the gateway device by using the general frame with service identifier being 0 (see 9.3);

– The network manager residing on the gateway device returns a join response command frame to the access device by using the general frame with service identifier being 1; subsequently, the network manager allocates communication resources (superframes, links, etc.) for the access device by using the general frame with service identifier being 12.

See 9.3 for the service identifiers.

The 8/16-bit short address of an access device (see 6.3) is unique in a WIA-FA network.

9.3 Frame formats between GW and AD

The general format of frames between the gateway device and an access device is shown in Figure 69.

1 octet	0/1 octet	0/8 octet(s)	2 octets	Variable length
Service identifier	AdID	AD long address	Length of service parameters	Service parameters

Figure 69 – General frame format between GW and AD

Each field in Figure 69 is defined as follows:

– Service identifier: the length is 1 octet. Service identifier is used to identify different services (see Table 68) between the gateway device and an access device.

– AdID: the length is 1 octet. AdID is used to identify ADs (see Table 19). AdID is invalid if service identifier is 0 or 1;

– AD long address: the length is 0 or 8 octet(s). If the service identifier is 0 or 1, AD long address is the EUI-64 of an access device; otherwise, AD long address is invalid;

– Length of service parameters: the length is 2 octets.

– Service parameters: the length is variable. Service parameters are wired service data between gateway device and an access device. See Table 68 for the definitions of Service parameters.

The services and service identifiers are defined in Table 68.

Table 68 – Wired services between GW and AD

Service identifiers	Services	Service Parameters	Service transition
0	AD join request	See Table 69	AD→GW
1	AD join response	See Table 70	GW→AD
2	GW request AD to send GACK	See Table 71 and Table 72	GW→AD
3	GW request AD to send NACK	See Table 73	GW→AD
4	Data request	Parameters defined in DLDE-DATA.request (see 8.2.2)	GW→AD
5	Data indication	Parameters defined in DLDE-DATA.indication (see 8.2.3)	AD→GW
6	Device join indication	Parameters defined in DLME-JOIN.indication (see 8.3.4.2)	AD→GW
7	Device join response	Parameters defined in DLME-JOIN.response (see 8.3.4.3)	GW→AD
8	Device status report	Parameters defined in DLME-DEVICE-STATUS.indication (see 8.3.5.2)	AD→GW
9	Channel condition report	Parameters defined in DLME-CHANNEL-CONDITION.indication (see 8.3.6.2)	AD→GW
10	Remote attribute get request	Parameters defined in DLME-INFO-GET.request (see 8.3.7.1)	GW→AD
11	Remote attribute get confirm	Parameters defined in DLME-INFO-GET.confirm (see 8.3.7.4)	AD→GW
12	Remote attribute set request	Parameters defined in DLME-INFO-SET.request (see 8.3.8.1)	GW→AD
13	Remote attribute set confirm	Parameters defined in DLME-INFO-SET.confirm (see 8.3.8.4)	AD→GW
14	Device leave request	Parameters defined in DLME-LEAVE.request (see 8.3.9.1)	GW→AD
15	Key establish request	Parameters defined in KEY-ESTABLISH.request (see 11.2.2.1)	GW→AD
16	Key establish confirm	Parameters defined in KEY-ESTABLISH.confirm (see 11.2.2.4)	AD→GW
17	Key update request	Parameters defined in KEY-UPDATE.request (see 11.2.3.1)	GW→AD
18	Key update confirm	Parameters defined in KEY-UPDATE.confirm (see 11.2.3.4)	AD→GW
19	Security alarm indication	Parameters defined in SEC-ALARM.request (see 11.2.4.2)	AD → GW
20 to 255	Reserved		

The services identified by 0 and 1 are used for ADs joining WIA-FA network. The service parameters of AD join request and AD join response are shown in Table 69 and Table 70, respectively.

The services identified by 2 and 3 are used by GW to request ADs sending GACKs or NACKs. The related service parameters are shown in Table 71, Table 72, and Table 73 respectively.

Table 69 – Service parameters of AD join request

Parameter	Data type	Valid range	Description
NetworkID	Unsigned8	0 to 255	Network identifier, used for multiple networks coexisting (see NetworkID in 6.7.1.2.1)
PhyAddr	Unsigned64	0 to $(2^{64}-1)$	Long address of the new access device waiting to join (see LongAddress in 6.7.1.2.1)

Table 70 – Service parameters of AD join response

Parameter	Data type	Valid range	Description

Status	Unsigned8	0 to 255	Result of join request:
			0= SUCCESS;
			1= NetworkID mismatched;
			2= authentication failure;
			3= network overload;
			others are reserved.

AdID	Unsigned	0 to 255	Identifier of an access device allocated by the gateway device (see AdID in 6.7.1.2.1). AdID is valid if Status is SUCCESS
ADAddr	Unsigned8/ Unsigned16	0 to 255/ 0 to 65 535	The 8/16-bit short address of an access device allocated by the gateway device

Table 71 – Service parameters of GW requesting AD to send GACK

Parameter	Data type	Valid range	Description
DeviceCount	Unsigned8	0 to 255	Count of field devices received frames by any access device
GACKInformation	GACKInfo_Struct List		Information of DeviceCount field devices, indicating by GACKInfo_Struct (see Table 72)

Table 72 – Parameters of GACKInfo_Struct structure

Parameter	Data type	Valid range	Description
DstAddr	Unsigned8/ Unsigned16/ Unsigned64	0 to 255/ 0 to 65 535/ 0 to $(2^{64}-1)$	Destination address, which can be short address or long address.
SequenceNumber	Unsigned16	0 to 65 535	Sequence number of a received frame

Table 73 – Service parameters of GW requesting AD to send NACK

Parameter	Data type	Valid range	Description
RetryDeviceCount	Unsigned8	0 to 255	Count of field devices required retransmission
DstAddressList	Unsigned8/ Unsigned16		Destination address. See DeviceShortAddress in 6.7.1.2.1

10 Application Layer

10.1 Overview

WIA-FA application layer (AL) provides distributed applications for users. It defines application objects that interact with industrial processes as well as communication services that support communications among distributed applications. AL is comprised of UAPs and

ASL. Each UAP is composed of one or more UAOs. DMAP is a special UAP (see 6.2). ASL defines communications among UAPs on different devices.

10.2 AL protocol stack

Figure 70 shows the position and construction of AL within the WIA-FA communication protocol stack, and the grey parts denote the AL related portions.

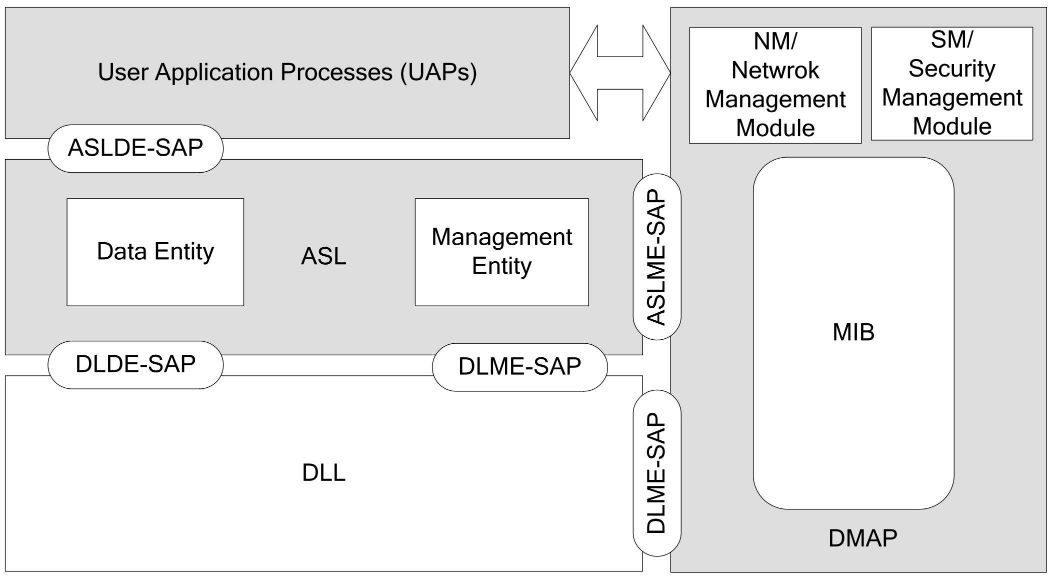
User Application Processes (UAPs)
ASLDE-SAP
Data Entity
ASL
Management Entity
ASLME-SAP
DLDE-SAP
DLME-SAP
DLL
DLME-SAP
NM/ Netwrok Management Module
SM/ Security Management Module
MIB
DMAP

Figure 70 – AL portions within WIA-FA protocol stack

The functions of WIA-FA ASL are performed by ASL data entity (ASLDE) and ASL management entity (ASLME). ASLDE performs ASL data transmission function (see 10.7.2) and the data service interface is provided for UAPs on ASLDE-SAPs. ASLME performs application configuration function (see 10.5.5) and the management service interface is provided on ASLME-SAPs.

10.3 AL functions

10.3.1 Data function

Three types of application data are transferred between the gateway device and field devices, including periodical measurement values (i.e. input data) and set point values (i.e. output data), requests and responses of attribute read and write access, and alarm reports in abnormal cases. WIA-FA defines corresponding application services (see 10.6) and virtual communication relationships (see 10.5.5.3) to support the usages and transmissions for the different application data.

The maximum length of WIA-FA application service messages is restricted by DLL resources and transmission capability. The maximum length of DLL payload is specified by MaxPayloadLength attribute in MIB (see 6.7.1.2.1).

10.3.2 Management function

The input and output data of field devices shall be transferred periodically according to predefined cycle. When a field device joins WIA-FA network, it shall be configured by the host computer, i.e. specifying the UAOs of the field device to be used, the input data and output data of UAOs to be transferred periodically, and the corresponding data update rate (see 10.5.5.2 for UAO configuration).

If multiple UAOs are assigned to one UAP on a field device, these UAOs shall have the same data update rate and use the same P/S VCR.

10.3.3 Communication mode

ASL supports three communication models: Client/Server (C/S) model, Publisher/Subscriber (P/S) model and Report source/Sink (R/S) model. These communication models are used to transmit the application data with corresponding priority.

– C/S communication model: applicable to aperiodic non-real-time read/write access and alarm acknowledgement (NRT). The transmission is unicast.

– P/S communication model: applicable to periodic process data publishment (RT1). The transmission can be unicast or broadcast.

– R/S communication model: applicable to aperiodic alarm reports (RT2) or urgent command (RT0). The transmission can be unicast or broadcast.

The roles played by the gateway device and field devices in different communication models are shown in Table 74.

Table 74 – Communication models between gateway device and field devices

Gateway Device	Field Device	Communication Mode	Priority	Usage
Client	Server	unicast	NRT	Gateway device reads or writes the attributes of MIB or UAOs, or acknowledge the alarms for field devices.
Publisher	Subscriber	unicast	RT1	Gateway device publishes the output data to a field device.
		broadcast	RT1	Gateway device publishes the output data to all field devices.
Subscriber	Publisher	unicast	RT1	Field device publishes the input data to gateway device.
Report Sink	Report Source	unicast	RT2	Field device reports alarms to the gateway device.
Report Source	Report Sink	unicast	RT0	Gateway device sends the start or stop command to a field device.
		broadcast	RT0	Gateway device sends the start or stop command to all field devices.

These three communication models are accomplished by corresponding types of VCRs. VCR defines the logic communication relationship between gateway device and field device, which is represented as the VCR endpoint in the devices. VCR endpoint defines the communication related attributes of VCR and is identified uniquely by VCR_ID in the device.

10.4 Application data

10.4.1 General

In WIA-FA devices, the application data which is used for UAOs includes attribute data, process data and event data. Attribute data are the attributes for aperiodic read and write accesses, including structured and unstructured attributes in MIB (see 6.7.1.2) as well as the attributes associated with the process or technology in UAOs. Process data is the input and output data of UAOs, which is periodically exchanged between field devices and the gateway device. Event data is the alarm reported by field devices to the gateway device.

10.4.2 Process data

10.4.2.1 General

Process data that is periodically transferred between field devices and gateway device includes input data and output data. Input data is the data such as sensor measurement value or actuator feedback value, and output data is the data such as actuator set point value. Process data are also divided into analog data and digital data. Such data can be values (e.g. float or unsigned type) or values with status. Manufacturer specific process data is allowed in WIA-FA.

10.4.2.2 Analog data with status

WIA-FA defines two types of analog data with status, i.e. SingleAnalogData and DoubleAnalogData, as shown in Table 75 and Table 76.

Table 75 – SingleAnalogData definition

MemberID	Member name	Data type	Valid range	Description
0	Value	Single Float		Value of the data

1	Status	Unsigned8	0 to2	Status of the data, and the value shall be:
				0 = GOOD;
				1 = BAD;
				2 = UNCERTAIN;
				Others are reserved.

Table 76 – DoubleAnalogData definition

MemberID	Member name	Data type	Valid range	Description
0	Value	Double Float		Value of the data

1	Status	Unsigned8	0 to2	Status of the data, and the value shall be:
				0 = GOOD;
				1 = BAD;
				2 = UNCERTAIN;
				Others are reserved.

10.4.2.3 Digital data with status

WIA-FA defines three types of digital data with status, i.e. DigitalData8, DigitalData16 and DigitalData32, as shown in Table 77, Table 78 and Table 79.

Table 77 – DigitalData8 definition

MemberID	Member name	Data type	Valid range	Description
0	Value	Unsigned8		Value of the data

1	Status	Unsigned8		Status of the digital data, and the value shall be:
				0 = GOOD;
				1 = BAD;
				2 = UNCERTAIN;
				others are reserved

Table 78 – DigitalData16 definition

MemberID	Parameter name	Data type	Valid range	Description
0	Value	Unsigned16		Value of the data
1	Status	Unsigned8		Status of the data, and the value shall be: 0 = GOOD 1 = BAD 2 = UNCERTAIN Others are reserved

Table 79 – DigitalData32 definition

MemberID	Member name	Data type	Valid range	Description
0	Value	Unsigned32		Value of the data
1	Status	Unsigned8		Status of the digital data, and the value shall be: 0 = GOOD 1 = BAD 2 = UNCERTAIN Others are reserved

10.4.3 Event data

WIA-FA defines EventData for field devices to maintain alarm events. When an abnormal event occurs, its EventFlag bit shall be set. When the abnormal event disappears, its EventFlag bit shall be reset. AckFlag is used to indicate whether this alarm event shall be acknowledged. EventData is defined in Table 80. The events defined for UAOs are shown in Table 81.

Table 80 – EventData Definition

MemberID	Member name	Data type	Data length	Valid range	Description
1	EventFlag	Bit Field	2		The coding for each bit is as follows: 0 = event doesn’t exist; 1 = event exists.
2	AckFlag	Bit Field	2		Flag indicating whether the corresponding event need to be acknowledged, The coding for each bit is as follows: 0 = acknowledgement not needed; 1 = acknowledgement needed.

Table 81 – UAO events definitions

Bit	Event Type
Bit 0	CONFIGURATION_ERROR
Bit 1	SENSOR_FAULT
Bit 2	ACTUATOR_FAULT
Bit 3	INPUT_EXCEEDS_UPPER_LIMIT
Bit 4	INPUT_EXCEEDS_LOWER_LIMIT
Bit 5	OUTPUT_EXCEEDS_UPPER_LIMIT
Bit 6	OUTPUT_EXCEEDS_LOWER_LIMIT
Bit 7	PROCESS_DATA_NOT_UPDATED
Bit 8	PROCESS_DATA_LENGTH_INCONSISTENT
Bit 9 to Bit15	Reserved
Bit 16 to Bit23	Manufacturer specific events

10.5 User application process

10.5.1 General

WIA-FA defines distributed application process (DAP) to accomplish the distributed applications in the industrial field environment. A DAP can locate on one or more than one devices over WIA-FA network. DAP is implemented as UAP on the device, which is identified by UAP Identifiers (UAP_ID) uniquely within WIA-FA devices. A WIA-FA device can support one or more UAPs. Figure 71 shows the relationships between DAPs and UAPs over WIA-FA network. DMAP is a special kind of UAP accomplishing system and security management functions (see 6.2). Each WIA-FA device shall implement only one DMAP. The UAP_ID of DMAP is 0.

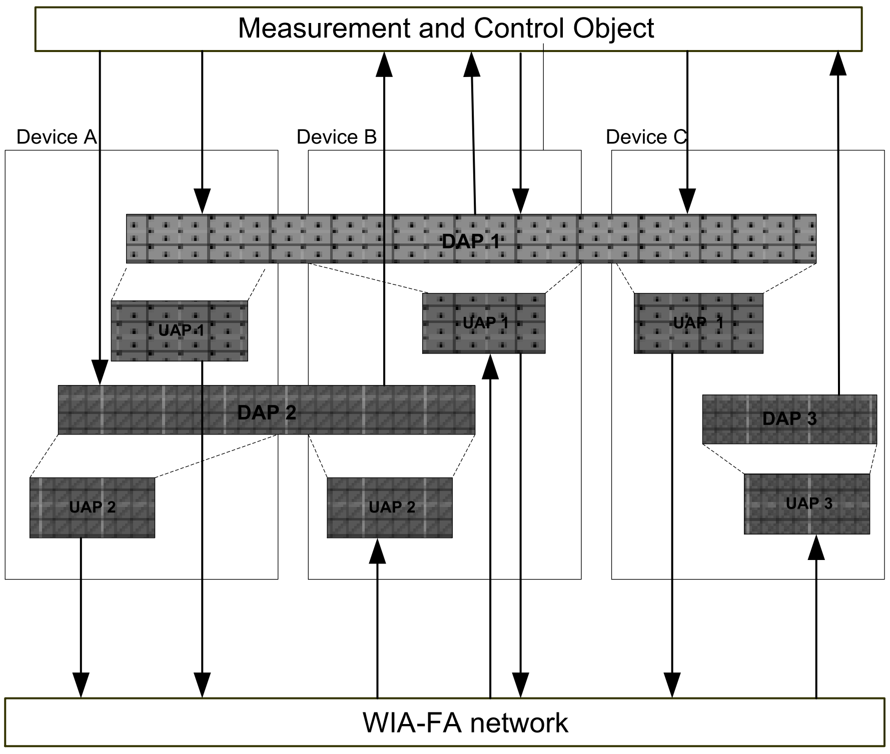

Figure 71 – The relationships between UAPs and DAPs

10.5.2 User application object

A UAP is composed of one or more UAO(s) within one field device, shown in Figure 72. Each UAO manages and provides the run time exchange of messages across the network and within the device.

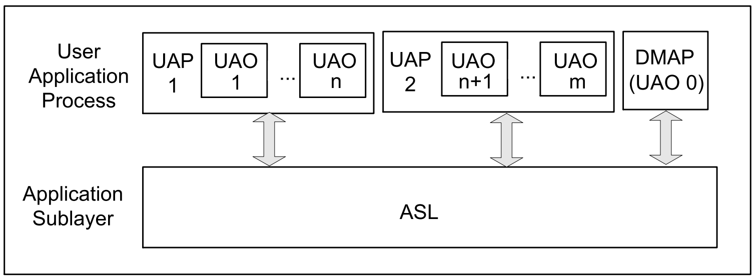

Figure 72 – User application objects

WIA-FA supports four types of UAOs: analog input (AI), analog output (AO), digital input (DI), and digital output (DO). Manufacturer specific UAOs are allowed in WIA-FA. The implementation of UAO types is optional for field devices. The definition of UAO attributes is beyond the scope of this PAS.

UAO shall be identified uniquely by UAP_ID within a device. DMAP is a special kind of UAP and has only one UAO (UAO_ID=0).

10.5.3 IO data images on gateway device

It is optional for gateway device to implement IO data images for field devices. After a field device joins WIA-FA network, the gateway device may allocate an IO data image for the field device, which is used to buffer the periodically transferred input and output data of the field device. Thus, when the gateway device interconnects with another control system (e.g. fieldbus), it can communicate with other devices over the control network as a remote I/O device.

The gateway device shall save the configuration data of the field device, which are used to interpret the input and output data of field device.

When a field device leaves WIA-FA network, the gateway device shall release the IO data image for the field device if implemented.

Figure 73 shows an implementation example of the IO data images on the gateway device. Implementation is manufacture specific and beyond the scope of this PAS.

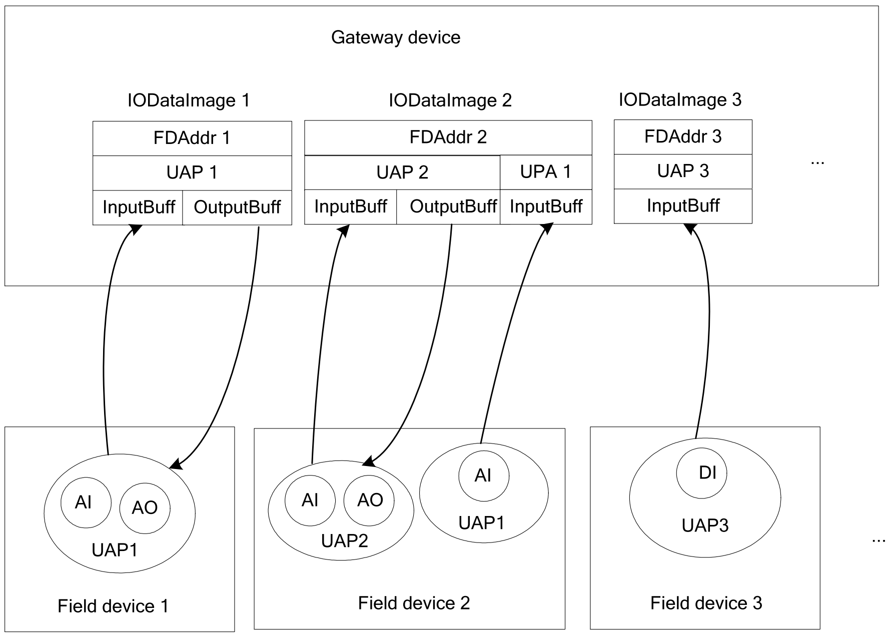

Figure 73 – Implementation example of IO data images on the gateway device

10.5.4 Alarm mechanism

All UAOs in the field device should maintain an event data (EventData type, see Table 80). When an abnormal event (alarm event) appears or disappears, the corresponding EventFlag bit shall be set (appear) or reset (disappear) The AckFlag bits of the event data indicate whether the corresponding events shall be acknowledged by the gateway device, which can be configured by the gateway device. UAP shall pack the event data into a REPORT request message and send it to the gateway device.

The gateway device shall maintain an alarm queue for each field device to save the received alarm events. Depending on the AckFlag (AckFlag = 1) indications of the event data, the host computer may return a REPORT ACK request message to the field device.

For those alarm events requiring acknowledgement, when UAP of the field device receives the REPORT ACK request message, it shall clear the alarm events of UAO that are reported previously, i.e. resetting the respective EventFlag bits. If the field device doesn’t receive the REPORT ACK request message within AlarmRptDur (see Table 14) time interval, UAP shall report the alarm events again. AlarmRptDur can be configured by the host computer. The acknowledge requirements of UAO alarm events is also configurable.

The field device can report more than one alarm event and the gateway device can acknowledge parts of or all alarms depending on the actual conditions.

10.5.5 Application configuration process

10.5.5.1 General

The application configuration process for the field device is divided into the following steps:

a) write CfgUAOList for setting application configuration data;

b) write VCRList for configuring the C/S VCR, P/S VCRs, and R/S VCRs.

10.5.5.2 UAO configuration

When a field device joins WIA-FA network, the gateway device shall first read its NumOfSupUAO (see Table 19) attribute to obtain the number of UAO types supported by the field device, and read SupUAOList (UAOClassDesc_Struct type, see Table 22) to obtain all UAO class descriptions which are implemented by the field device. The UAO class description includes Class_ID, UAO type, minimum data update rate, as well as the data type and length of the input data and output data supported by the UAO. The gateway device shall deliver this information to the host computer.

The host computer shall configure UAPs for the field device according to real application requirements, instantiate several UAOs of the UAO classes supported by the field device, and assign these UAOs to UAPs. All UAOs belonging to one UAP shall have the same data update rate.

The host computer shall configure NumOfCfgUAO (see Table 19) and CfgUAOList (UAOInstDesc_Struct, see Table 24) to the field device. The sum of all the input/output data length of a configured UAO shall be less than the maximum input/output data length specified by the UAO class. The type and order of the periodically transferred input/output data shall be consistent with the configuration.

When a field device leaves WIA-FA network, it shall clear the CfgUAOList.

10.5.5.3 VCR Configuration

After configuring UAOs, the gateway device shall configure VCRs for the field device according to the UAO configuration. The host computer shall write VCRList (see Table 15) to the field device for setting communication related attributes of P/S VCRs, and R/S VCRs. There is a default C/S VCR (VCR_ID = 0) between the gateway device and a field device. The configuration of C/S VCR for a field device is optional.

Table 82 shows the overview of attribute settings for different VCR type on a field device. See Table 21 for VcrEP_Structure definition.

Table 82 – VCR attribute configuration overview

MemberID	**Member name**	**VcrEP_Type**			
		SERVER	**PUBLISHER**	**SUBSCRIBER**	**REPORT SOURCE**
0	VCR_ID	0	Configured	Configured	Configured
1	VcrEP_Type	1	2	3	4
2	UAP_ID	0	Configured	Configured	Configured
3	PeerAddr	Short address of gateway device	Short address of gateway device	Short address of gateway device	Short address of gateway device
5	VCRActiveTime	Invalid and should be set 0	0 or configured	0 or configured	Invalid and should be set 0
6	DataUpdateRate	Invalid and should be set 0	configured	configured	Invalid and should be set 0
7	Deadline	Invalid and should be set 0	configured	configured	Invalid and should be set 0
8	WatchdogTime	Default value is 100 ms, and it could be changed by host computer	Invalid and should be set 0	Invalid and should be set 0	Invalid and should be set 0

When configuring the VCRList for a field device, the gateway device shall create the corresponding VCR endpoints in order to establish the VCR connections with the field device.

When the field device leaves WIA-FA network, it shall clear the VCRList except the server VCR endpoint.

10.5.5.3.1 C/S VCR configuration

Only one default C/S VCR (VCR_ID = 0) is needed between the gateway device and a field device. The gateway device is client while the field device is server.

Figure 74 shows the C/S VCR relationship between the gateway device and field devices. Arrows indicate the data transmission directions.

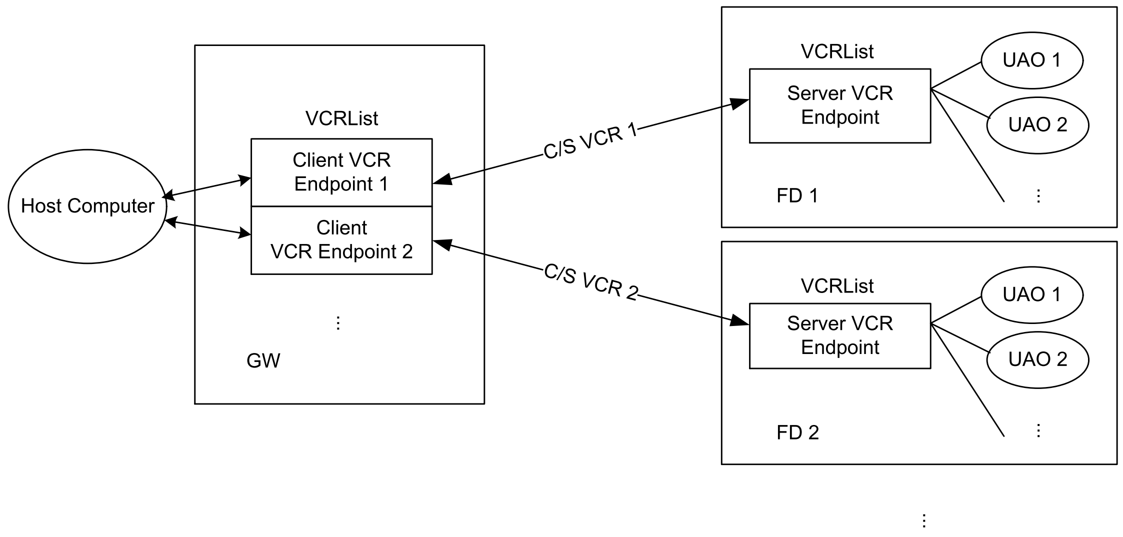

Figure 74 – C/S VCR relationships between GW and FDs

10.5.5.3.2 P/S VCR configuration

Multiple UAOs may be allocated to one UAP on a field device. If the UAOs have input data, a publisher VCR endpoint shall be configured and the field device acts as a publisher. If the UAOs have output data, a subscriber VCR endpoint shall be configured and the field device acts as a subscriber. Each VCR endpoint should be allocated a buffer to store the input or output data.

Figure 75 and Figure 76 show the P/S VCR relationships between the gateway device and field devices. The arrows indicate the data transmission directions.

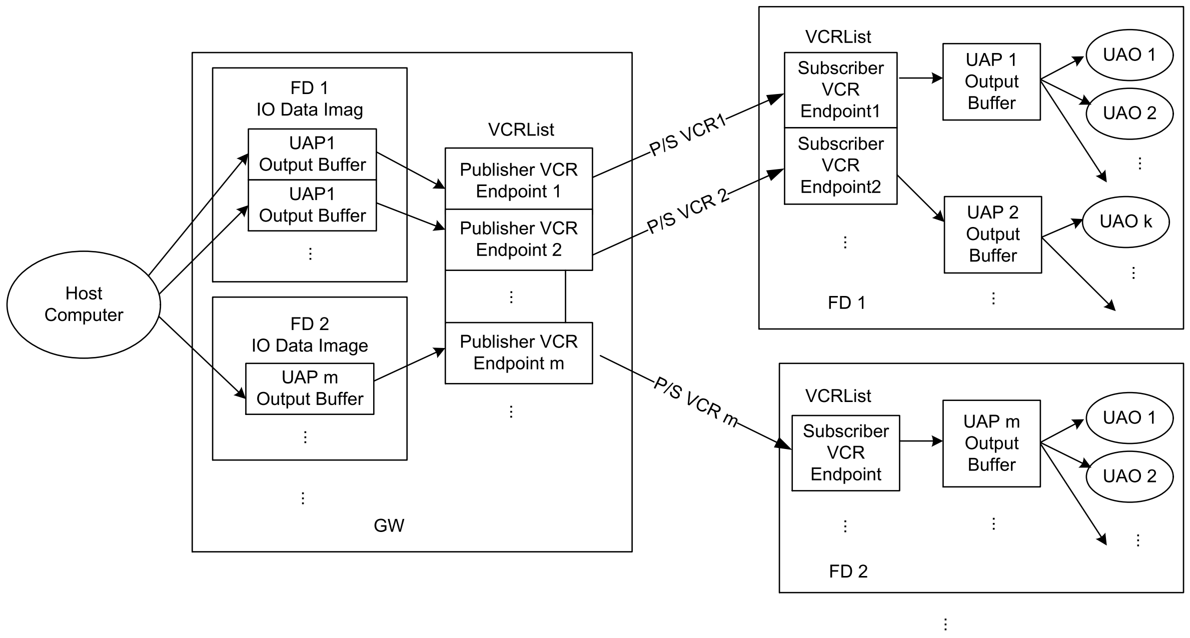

Figure 75 – P/S VCR relationships between GW and FDs

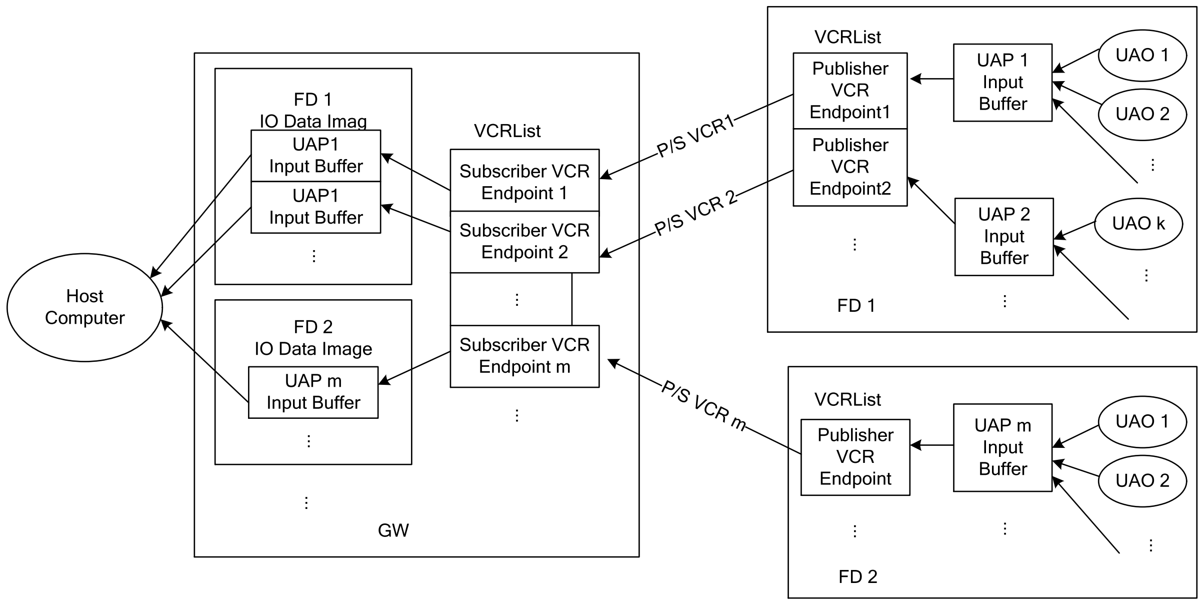

Figure 76 – P/S VCR relationships between FDs and GW

10.5.5.3.3 R/S VCR configuration

At least one P/S VCR shall be established between the gateway device and a field device. The field device is report source while the gateway device is report sink.

Figure 77 shows the R/S VCR relationship between the gateway device and field devices. Arrows indicate the data transmission directions.

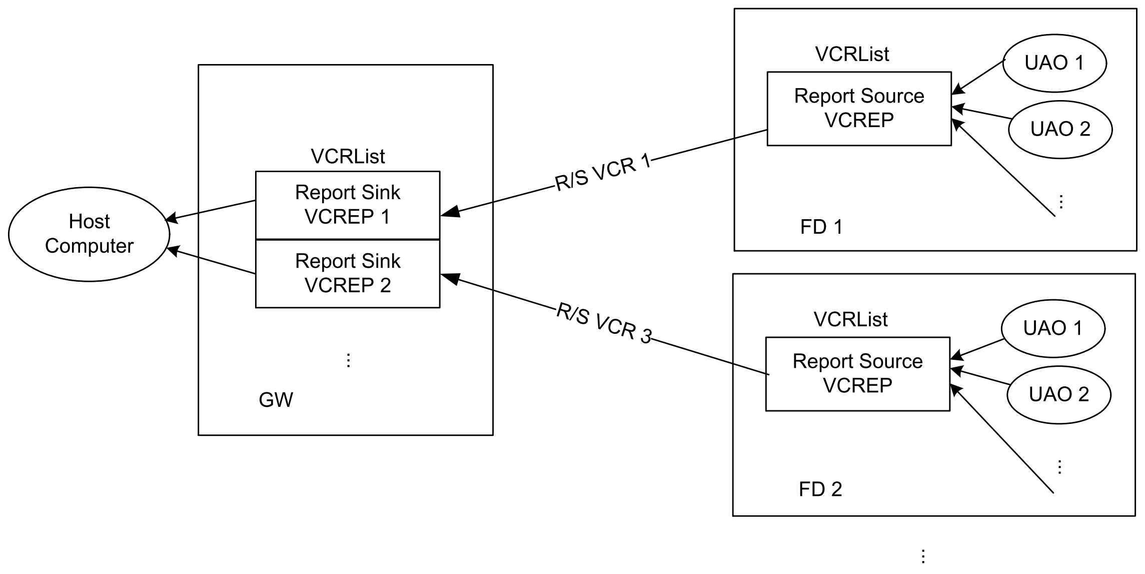

Figure 77 – R/S VCR relationships between FDs and GW

10.5.5.4 Configuration process

Figure 78 shows the configuration process from the gateway device to a field device.

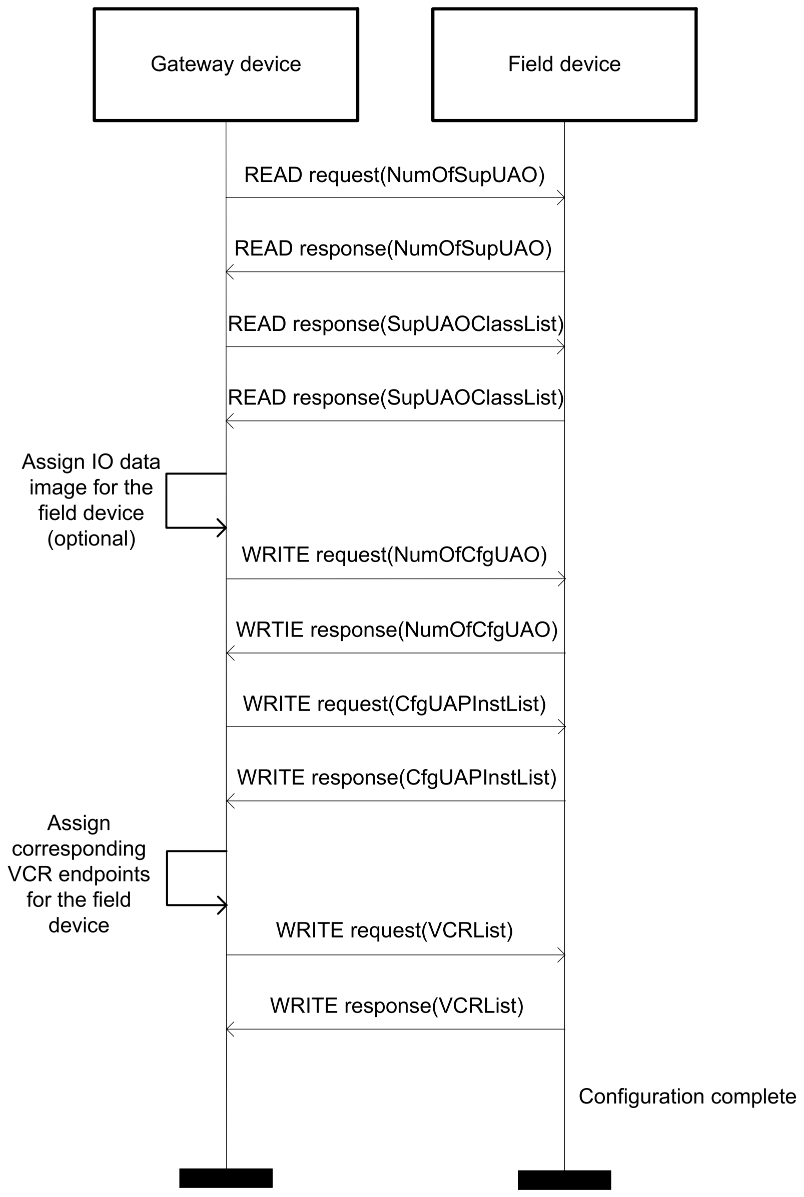
Gateway device
Field device
READ request(NumOfSupUAO)
READ response(NumOfSupUAO)
READ response(SupUAOClassList)
READ response(SupUAOClassList)
Assign IO data image for the field device (optional)
WRITE request(NumOfCfgUAO)
WRTIE response(NumOfCfgUAO)
WRITE request(CfgUAPInstList)
WRITE response(CfgUAPInstList)
Assign corresponding VCR endpoints for the field device
WRITE request(VCRList)
WRITE response(VCRList)
Configuration complete

Figure 78 – Configuration process for a field device

10.5.5.5 Aggregation and disaggregation of process data

Data of all UAOs allocated to one UAP shall be aggregated on the field device. As a publisher, the UAP shall obtain all process data according the configured order and form one PUBLISH request message. This is the AL aggregation process.

As a subscriber, the UAP shall parse the received PUBLISH request message and transfer the data to the respective UAOs. This is the AL disaggregation process.

Figure 79 shows an example of an UAO aggregation and disaggregation process.

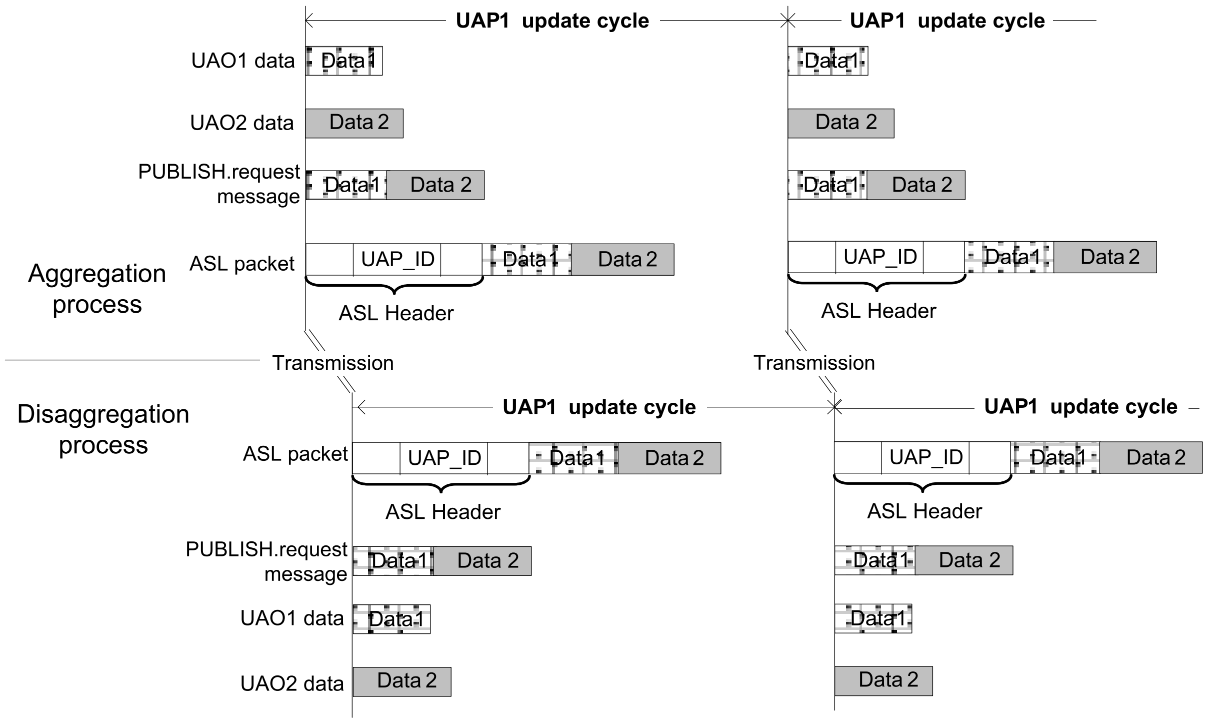

Figure 79 – UAO aggregation and disaggregation process

10.6 Application services

10.6.1 Confirmed services and unconfirmed services

WIA-FA application layer defines application services to support aperiodic attribute read/write access, periodic process data publishment, and the alarm event reports, shown as in Table 83.

The application services include confirmed services and unconfirmed services. Confirmed services are used for bidirectional request/response between UAPs, and the unconfirmed services are used for unidirectional data transmission from one UAP to one or more UAPs.

Table 83 – Application services supported by UAPs

Service Name	Service Identifier	Message Type	Description
READ	0x01	request	Request to read the value of a UAO or MIB attribute
		response(+)	The UAO or MIB attribute is read successfully, and the value of the attribute shall be returned in the response
		response(-)	Fail to read the UAO or MIB attribute, and the failure reason shall be returned in the response
WRITE	0x02	request	Request to write the value of a UAO or MIB attribute
		response(+)	The UAO or MIB attribute is written successfully,
		response(-)	Fail to write the UAO or MIB attribute, and the failure reason shall be returned in the response
PUBLISH	0x03	request	Request to publish the input or output process data
REPORT	0x04	request	Request to report a or multiple UAO event(s)
REPORT ACK	0x05	request	Acknowledge the reported alarm(s)
		response(+)	Alarm(s) is/are acknowledged successfully
		response(-)	Fail to acknowledge the alarms, and the failure reason shall be returned in the response

10.6.2 READ service

10.6.2.1 Message format

Figure 80 shows the format of READ request message.

2 octets	1 octet	1 octet	1 octet
UAO Identifier	Attribute Identifier	Storage Index	Member Identifier

Figure 80 – READ request message format

Figure 81 shows the format of READ response(+) message.

Variable length in octet
Data

Figure 81 – READ response(+) message format

Figure 82 shows the format of READ response(-) message.

1 octet	1 octet
Error code	Additional information

Figure 82 – READ response(-) message format

The fields of the READ messages are described as follows:

- UAO Identifier: UAO_ID of the UAO that is read, value 0 is used for MIB;
- Attribute Identifier: AttributeID of the attribute in UAO or MIB;
- Storage Index: Index of a record in the list. Value 255 indicates all records shall be read. This field is invalid for the unstructured attributes;
- Member Identifier: MemberID of a member in a structured attribute. Value 255 indicates all members shall be read. This field is invalid for the unstructured attributes;
- Data: Value that is read;
- Error Code: Failure reason code. The definition is shown in Table 84;
- Additional Information: Additional information for failure reason, manufacturer specific.

Table 84 – Error code definition for READ response(-) message

Values	Meaning
1	SERVICE_EXPIRATION
2	SERVICE_NOT_SUPPORTED
3	UAO_NOT_EXISTENT
4	ATTRIBUTE_NOT_EXISTENT
5	STOREINDEX_NOT_EXISTENT
6	MEMBER_NOT_EXISTENT
7	LENGTH_TOO_LARGE [a]
8	OTHERS
9 to 255	Reserved for future use

[a] MaxPayLoadLenth describes the max length of DLL payload, when the size of Data is more than MaxPayLoadLength, this error code is returned.

10.6.2.2 Service process

This confirmed service transmits NRT data via C/S VCR. The gateway device shall use this service to read aperiodically an attribute or an attribute member of a UAO. If the UAP_ID, AttributeID or MemberID is wrong, or the service is not supported by the field device, it shall return the appropriate error code. Figure 83 shows the READ service process.

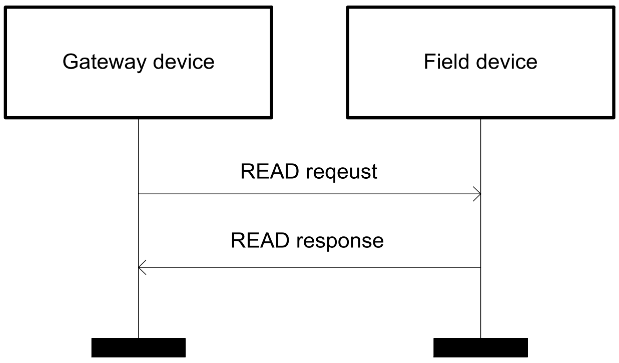

Figure 83 – READ Service process

10.6.3 WRITE service

10.6.3.1 Message format

Figure 84 shows the format of WRITE request message.

2 octets	1 octet	1 octet	1 octet	Variable length
UAO Identifier	Attribute Identifier	Storage Index	Member Identifier	Data

Figure 84 – WRITE request message format

No format is defined for WRITE response(+) message.

Figure 85 shows the format of WRITE response(-) message.

1 octet	1 octet
Error code	Additional information

Figure 85 – WRITE response(-) message format

The fields of the WRITE messages are described as follows:

– UAO Identifier: UAO_ID of the UAO that is written, value 0 is used for MIB;

– Attribute Identifier: AttributeID of the attribute in UAO or MIB;

– Storage Index: Index of a record in the list. Value 255 indicates all records shall be written. This field is invalid for the unstructured attributes;

– Member Identifier: MemberID of a member in a structured attribute. Value 255 indicates all members shall be written. This field is invalid for the unstructured attributes;

– Data: value that is written;

– Error Code: Failure reason code. The definition is shown in Table 85;

– Additional Information: Additional information for failure reason, manufacturer specific.

Table 85 – Error code definition for WRITE response(-)

Values	Meaning
1	SERVICE_EXPIRATION
2	SERVICE_NOT_SUPPORTED
3	UAO_NOT_EXISTENT
4	ATTRIBUTE_NOT_EXISTENT
5	STOREINDEX_NOT_EXISTENT
6	MEMBER_NOT_EXISTENT
7	LENGTH_NOT_MATCH
8	VALUE_EXCEED_SCOPE
9	OTHERS
10 to 255	Reserved for future use

10.6.3.2 Service process

This confirmed service transmits NRT data via C/S VCR. The gateway device shall use this service to write an attribute or an attribute member of a UAO or MIB aperiodically. If the UAP_ID, AttributeID or MemberID is wrong, or the value exceeds scope, or the data length does not match, or the service is not supported by the field device, it shall return the appropriate error code. Figure 86 shows the WRITE service process.

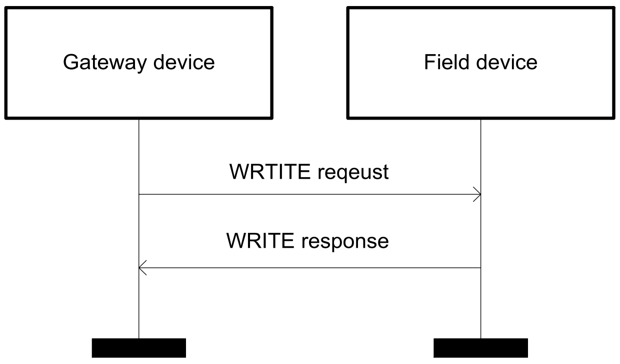

Figure 86 – WRITE Service process

10.6.4 PUBLISH Service

10.6.4.1 Message format

Figure 87 shows the format of the PUBLISH request message.

Variable Length in octet
Data

Figure 87 – PUBLISH request message format

PUBLISH service is an unconfirmed service and has no response.

The fields of the PUBLISH request message are described as follows:

– Data: all input and output data that is published by UAP.

10.6.4.2 Service process

This unconfirmed service transmits RT1 data via P/S VCR. The gateway device or field devices shall use this service to publish periodically the process data upon DataUpdateRate. UAOs belonging to the same UAP shall be aggregated to one PUBLISH service. Figure 88 shows the PUBLISH service process from the field device to the gateway and Figure 89 shows the opposite direction.

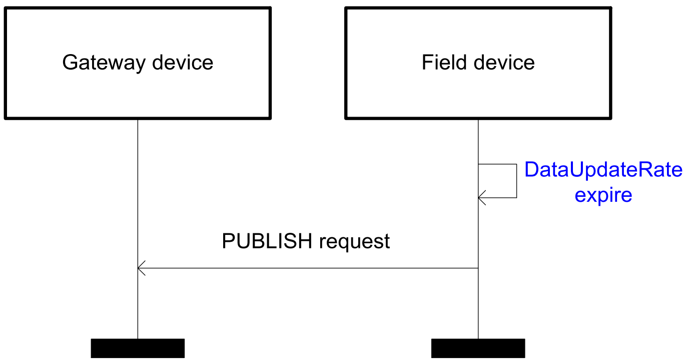

Figure 88 – PUBLISH Procedure from Field Device to Gateway Device

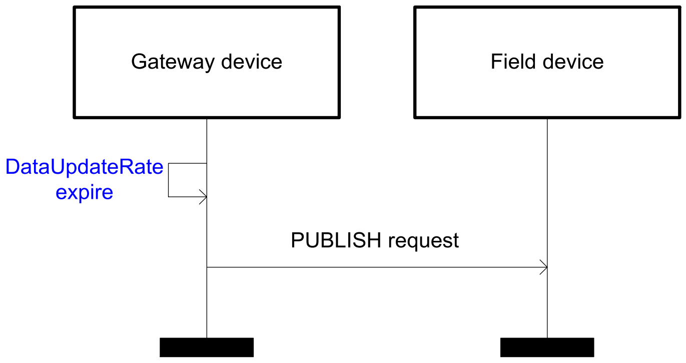

Figure 89 – PUBLISH Procedure from Gateway Device to Field Device

10.6.5 REPORT Service

10.6.5.1 Message format

Figure 90 shows the format of the REPORT request message.

1 octet	4 octets	1 octet
UAO_ID	Event	Additional information

Figure 90 – REPORT request message format

REPORT service is an unconfirmed service and has no response.

The fields of the REPORT request message are described as follows:

– UAO_ID: UAO_ID of the UAO that reports the alarm events;

– Event: the reported alarm events with the data type of EventData (see Table 80);

– Additional information: manufacturer specific additional information.

This unconfirmed service transmits RT0 data via R/S VCR. The field device shall use this service to report the appearance or disappearance of one or more than one alarm event to the gateway device. Figure 91 shows the REPORT service process.

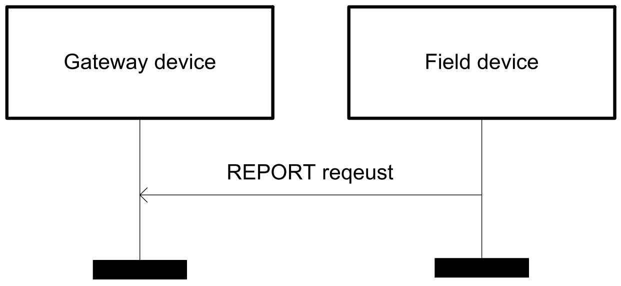

Figure 91 – REPORT Service process

10.6.6 REPORT ACK

10.6.6.1 Message format

Figure 92 shows the format of the REPORT ACK request message.

1 octet	2 octets
UAO_ID	AckEvent

Figure 92 – REPORT ACK request message format

Figure 93 shows the format of the REPORT ACK response(+) message.

1 octet
UAO_ID

Figure 93 – REPORT ACK response(+) message format

Figure 94 shows the format of the REPORT ACK response(-) message.

1 octet	1 octet	1 octet
UAO_ID	Error Code	Additional information

Figure 94 – REPORT ACK response(-) message format

The fields of the REPORT ACK messages are described as follows:

– UAO_ID: UAO_ID of the UAO that is alarm acknowledged;
– AckEvent: acknowledged events. Each bit represents one event defined in Table 81 and the value 1 indicates the corresponding event is acknowledged;
– Error Code: failure reason code. See Table 86 for definition;
– Additional information: manufacturer specific additional information.

Table 86 – Error code definition for REPORT ACK negative response

Values	Meaning
1	SERVICE_EXPIRATION
2	SERVICE_NOT_SUPPORTED
3	EVENT_NOT_EXISTENT
4	ACKNOWLEDGEMENT_NOT_REQUIRED
5	OTHER
6 to 255	Reserved for future use

10.6.6.2 Service process

As a confirmed service, this service transmits NRT data via C/S VCR. The gateway device shall use this service to acknowledge all or part of the events that have been reported previously from the field device. Figure 95 shows the REPORT ACK service process.

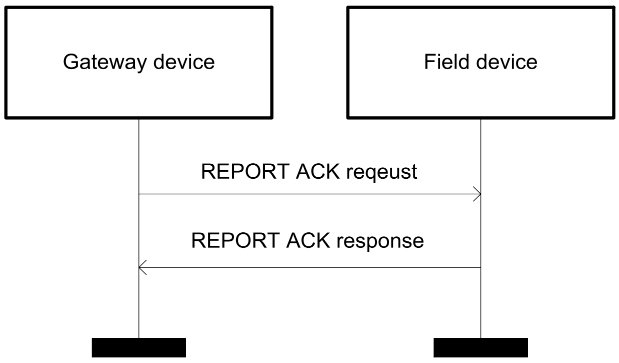

Figure 95 – REPORT ACK Service process

10.7 Application sublayer

10.7.1 Overview

Application sublayer (ASL) provides the transparent end-to-end data transmission service for the UAPs and DMAP.

10.7.2 ASL data service

10.7.2.1 General

ASL provides data service for UAPs in order to exchange application data between UAPs over WIA-FA network.

The ASL data service primitives include ASLDE-DATA.request, ASLDE-DATA.indication, ASLDE-DATA.response, and ASLDE-DATA.confirm.

10.7.2.2 ASLDE-DATA.request primitive

UAP transfers the application service request message to ASL by invoking ASLDE-DATA.request primitive. ASL shall form an APDU by adding ASL header according to ASL general packet format (see 10.7.3.1) and send it to DLL.

The definition of ASLDE-DATA.request primitive is as follows:

ASLDE-DATA.request(

```
          DstAddr,
          ServiceID,
          UAP_ID,
          Priority,
          AsduLength,
          Asdu
         )
```

The parameters of ASLDE-DATA.request primitive is described in Table 87.

Table 87 – ASLDE-DATA.request primitive parameter definitions

Parameter	Data type	Valid range	Description
DstAddr	Unsigned16	0 to 65 535	Destination address of the application service request
ServiceID	Unsigned8	1 to 5	Service identifier of the application service, see Table 83
UAP_ID	Unsigned8	0 to 255	Identifier of the UAP on the field device

Priority	Unsigned8	0 to 255	Priority of the application data, the values are as follows:
			0 = RT0;
			1= RT1;
			2 = RT2;
			4= NRT;
			other values are reserved. See 6.4.2.1 for priority details

AsduLength	Unsigned16	0 to 65 535	Length of the application service message
Asdu	Octetstring		Application service message

10.7.2.3 ASLDE-DATA.indication primitive

When ASL receives an application service request included in an ASL packet from DLL, it shall invoke ASLDE-DATA.indication primitive to transfer the request message to UAP.

The definition of ASLDE-DATA.indication primitive is as follows:

```
ASLDE-DATA.indication(
           ServiceID,
           UAP_ID,
           AsduLength,
           Asdu
          )
```

The parameters of ASLDE-DATA.indication primitive is described in Table 88.

Table 88 – ASLDE-DATA.indication primitive parameter definitions

Parameter	Data type	Value Range	Description
ServiceID	Unsigned8	0 to 255	Service identifier of the application service, see Table 83.
UAP_ID	Unsigned8	0 to 255	Identifier of the UAP on the field device
AsduLength	Unsigned16	0 to 65 535	Length of the application service message
Asdu	Octetstring		Application service message

10.7.2.4 ASLDE-DATA.response primitive

UAP transfers the confirmed application service response message to ASL by invoking ASLDE-DATA.response primitive. ASL shall form an APDU by adding ASL header according to ASL general packet format (see 10.7.3.1) and send it to DLL.

The definition of ASLDE-DATA.response primitive is as follows:

```
ASLDE-DATA.response(
            ServiceID,
            MsgType,
            UAP_ID,
            AsduLength,
            Asdu
     )
```

Parameters of ASLDE-DATA.response are described in Table 89.

Table 89 – ASLDE-DATA.response primitive parameter definition

Parameter	Data type	Value Range	Description
ServiceID	Unsigned8	0 to 255	Service identifier of the application service, see Table 83.

MsgType	Usigned8	0 to 255	Message type of the application service. The value shall be as follows:
			0 = REQUEST;
			1 = RESPONSE_P;
			2 = RESPONSE_N;
			Others are reserved.

UAP_ID	Unsigned8	0 to 255	Identifier of the UAP on the field device
AsduLength	Unsigned16	0 to 65 535	Length of the application service message
Asdu	Octetstring		Application service message

10.7.2.5 ASLDE-DATA.confirm primitive

When ASL receives a confirmed application service response included in an ASL packet from DLL, it shall invoke ASLDE-DATA.confirm primitive to transfer the response message to UAP.

The definition of ASLDE-DATA.confirm primitive is as follows:

```
ASLDE-DATA.confirm(
              SrcAddr,
              ServiceID,
              MsgType,
              UAP_ID,
              AsduLength,
              Asdu
             )
```

Parameters of ASLDE-DATA.confirm are described in Table 90.

Table 90 – ASLDE-DATA.confirmPrimitive Parameters

Parameter	Data type	Value Range	Description
SrcAddr	Unsigned16	0 to 65 535	Source address of the application service response
ServiceID	Unsigned8	0 to 255	Service identifier of the application service. See Table 83.

MsgType	Usigned8	0 to 255	Message type of the application service response. The value shall be as follows:
			1 = RESPONSE_P;
			2 = RESPONSE_N;
			Others are reserved.

UAP_ID	Unsigned8	0 to 255	Identifier of the UAP on the field device
AsduLength	Unsigned16	0 to 65 535	Length of the application service message
Asdu	Octetstring		Application service message

10.7.3 ASL packet format

10.7.3.1 General packet format

Each ASL packet is comprised of the following two parts:

– ASL Header, including Packet Control, UAP_ID, and Payload Length fields;

– ASL payload with variable length.

The ASL general packet format is shown in Figure 96.

ASL Header			ASL Payload
Packet Control	UAP Identifier	Payload Length	Payload
1 octet	1 octet	2 octets	Variable length

Figure 96 – ASL general packet format

10.7.3.2 ASL header

10.7.3.2.1 Packet Control field

10.7.3.2.1.1 General

The data type of Packet Control field is Unsigned8 with one-octet (8 bits) length, including ServiceID, Message Type subfield, shown in Figure 97.

Bit: 5 to 7	Bit: 3 to 4	Bit: 0 to 2
Reserved	Message Type	Service Identifier

Figure 97 – Format of packet control field

10.7.3.2.1.2 Service Identifier Subfield

The length of Service Identifier subfield is 3 bits, indicating the AL service types. Its values are shown in Table 91.

Table 91 – Service Identifier subfield definition

Bit: 0 to 2	Meaning
0b001	READ

0b010	WRITE
0b011	PUBLISH
0b100	REPORT
0b101	REPORT ACK

10.7.3.2.1.3 Message Type Subfield

The length of Message Type subfield is 2 bits, indicating the AL message types. Its values are shown in Table 92.

Table 92 – Message Type subfield definition

Bit: 3 to 4	Meaning
0b00	REQUEST
0b01	RESPONSE_P
0b10	RESPONSE_N
0b11	Reserved

10.7.3.2.2 UAP Identifer Field

The data type of UAP Identifier field is Unsigned8 with one octet length, and the value shall be UAP_ID indicating the UAP on the field device. UAP_ID = 0 is used for DMAP.

10.7.3.2.3 Payload Length Field

The data type of Payload Length field is Unsigned16 with 2-octet length, indicating the data length of ASL payload in octet, which excludes ASL header.

10.7.3.3 ASL payload

Payload field includes the AL service messages with variable length. Different services define different message formats; see 10.6.2 to 10.6.6 for details.

10.7.3.4 ASL state machine

According to the VCR endpoint types, the ASL state machines (ASLMs) include client state machine (AMCL), server state machine (AMSV), publisher state machine (AMPB), subscriber state machine (AMSB), report source state machine (AMRS), and report sink state machine (AMRK).

10.7.3.5 Primitives exchanged between ASL and UAP, DLL

10.7.3.5.1 Confirmed service primitives

Figure 98 and Table 93 show the primitives for confirmed service exchanged between ASL and general UAP, DLL.

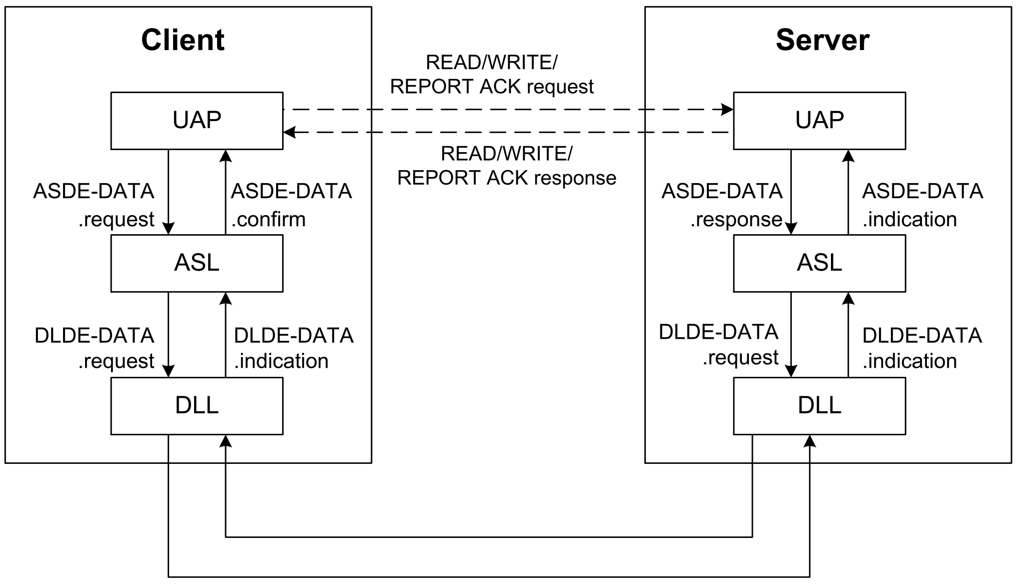

Figure 98 –Confirmed service primitives exchanged between layers

Table 93 –Confirmed service primitives exchanged between ASL and other layers

Primitives	Source	Parameters
ASLDE-DATA.request	UAP	DstAddr, ServiceID, UAP_ID, Priority, AsduLength, Asdu
ASLDE-DATA.indication	ASL	ServiceID, UAP_ID, AsduLength, Asdu
ASLDE-DATA.response	UAP	ServiceID, UAP_ID, AsduLength, Asdu
ASLDE-DATA.confirm	ASL	SrcAddr, ServiceID, UAP_ID, AsduLength, Asdu
DLDE-DATA.request	ASL	DstAddr, DataType, Priority, PayloadLength, Payload
DLDE-DATA.indication	DLL	SrcAddr, DataType, PayloadLength, Payload

10.7.3.5.2 Unconfirmed Service Primitives

Figure 99 and Table 94 show the primitives for unconfirmed service exchanged between ASL and general UAP/ DLL.

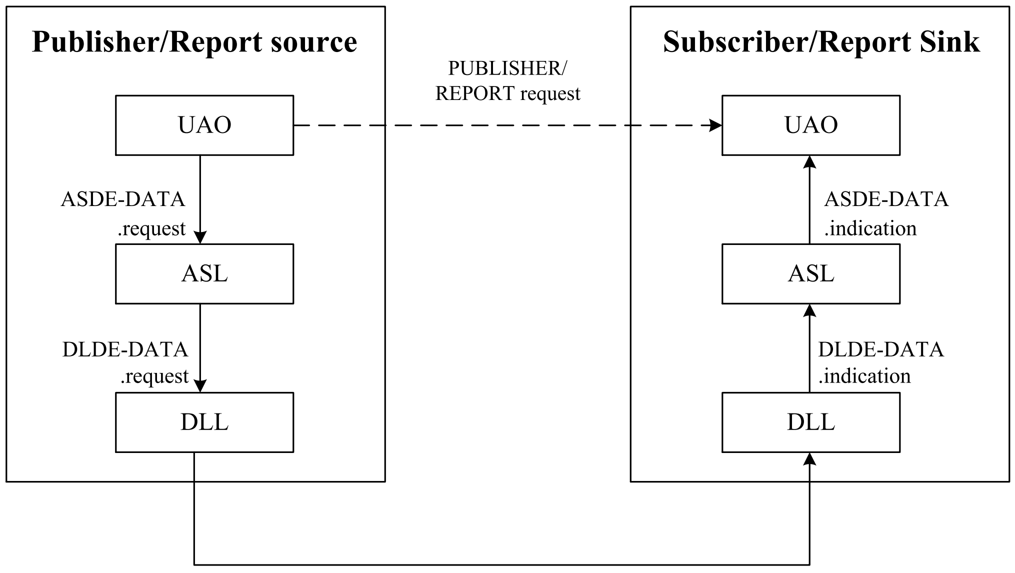

Figure 99 – Unconfirmed service primitives exchanged between layers

Table 94 – Unconfirmed service primitives exchanged between ASL and other layers

Primitives	Source	Parameters
ASLDE-DATA.request	UAP	DstAddr, ServiceID, UAP_ID, Priority, AsduLength, Asdu
ASLDE-DATA.indication	ASL	ServiceID, UAP_ID, AsduLength, Asdu
DLDE-DATA.request	ASL	DstAddr, DataType, Priority, PayloadLength, Payload
DLDE-DATA.indication	DLL	SrcAddr, DataType, PayloadLength, Payload

10.7.3.6 Primitives exchanged between ASL and DMAP/DLL

When the gateway device reads or writes the MIB attributes of a field device, the service primitives exchanged between ASL and DMAP, DLL are shown in Figure 100 and Table 95.

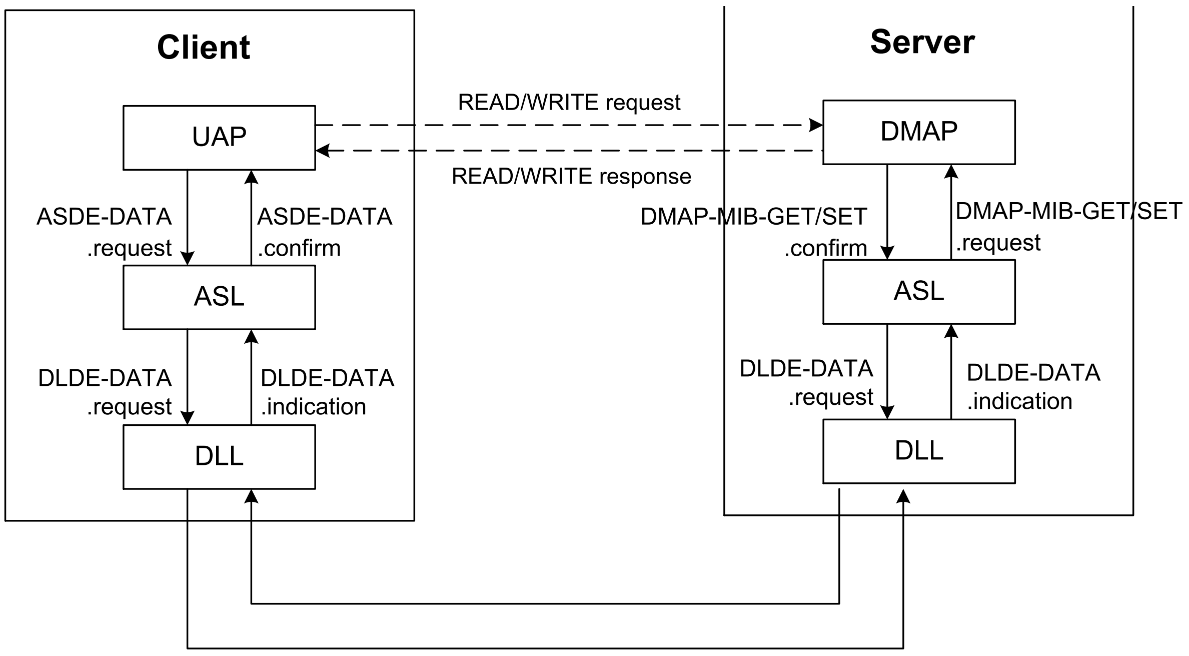

Figure 100 – Primitives invoking for read/ write MIB between layers

Table 95 – Primitives for read/ write MIB between layers

Primitives	Source	Parameters
ASLDE-DATA.request	UAP	DstAddr, ServiceID, VCR_ID, Priority, UAPSpec, AsduLength, Asdu
ASLDE-DATA.confirm	ASL	SrcAddr, ServiceID, AsduLength, Asdu
DMAP-MIB-GET.request	ASL	Handle, ShortAddress AttributeID, MemberID, FirstStoreIndex, Count
DMAP-MIB-SET.request	ASL	Handle, ShortAddress AttributeID, MemberID, FirstStoreIndex, Count, AttributeValue
DMAP-MIB-GET.confirm	DMAP	Handle, Status, Count, AttributeValue
DMAP-MIB-SET.confirm	DMAP	Handle, Status
DLDE-DATA.request	ASL	DstAddr, DataType, Priority, PayloadLength, Payload
DLDE-DATA.indication	DLL	SrcAddr, DataType, PayloadLength, Payload

10.7.3.7 Client state machine

The client state machine (AMCL) shall have the following states:

– Idle state: the initial and idle state of the client VCR endpoint. In this state, the VCR endpoint is waiting for ASLDE-DATA.request primitive to deliver a confirmed service request message. After receiving the primitive, it shall pack the request message into an ASL packet and invoke DLDE-DATA.request primitive to send the packet, and then transfer to Wait_Cnf state waiting for the corresponding service response.

– Wait_Cnf state: in this state, the client VCR endpoint is waiting for the AL service response returned from the field device. It shall perform one of the following state transitions:

  • If it receives the service response through DLDE-DATA.indication primitive within Watchdog interval, it shall parse the ASL packet and deliver the response message to UAP by invoking ASLDE-DATA.confirm primitive, and return to Idle state; or

- If it does not receive the service response within the Watchdog interval, it shall return a negative response message with “SERVICE_TIME_EXPIRATION” to UAP by invoking ASLDE-DATA.confirm primitive, and return to Idle state.

Figure 101 and Table 96 show the state transitions of the client state machine.

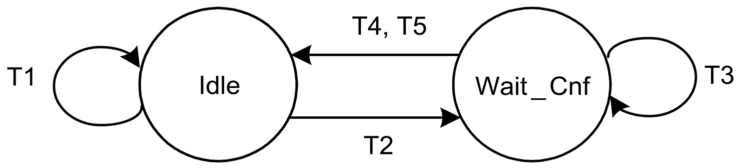

Figure 101 – State transition diagram of AMCL

Table 96 – State transition table of AMCL

#	Current State	Event or condition => action	Next state
T1	Idle	(ASLDE-DATA.request() && ServiceID != (READ || WRITE || REPORT ACK)) || ASLDE-DATA.response() || DLDE-DATA.indication() => Ignore;	Idle

T2	Idle	ASLDE-DATA.request() && ServiceID == (READ || WRITE || REPORT ACK)
		=>
		VCR_ID:= GetVcrID(DstAddr, CLIENT, UAP_ID);
		StartWatchdogTimer(VCR_ID);
		StoreSvrID(VCR_ID, ServiceID);
		MsgTyep:= REQUEST
		DLDE-DATA.request(
		    VCR_ID,
		    DataType:= DATA,
		    Priority,
		    PayloadLength:= AsduLength + 4,
		    Payload:= BuildAPDU(ServiceID, MsgTyep, UAP_ID, AsduLength , Asdu)
		)
			Wait_Cnf

T3	Wait_Cnf	ASLDE-DATA.request()
		|| ASLDE-DATA.response()
		|| (DLDE-DATA.indication()
		  && (DataType != DATA || TakeServiceID(Payload) != (READ||WRITE||REPORT ACK)
		  || TakeMsgType(Payload) != (RESPONSE_P || RESPONSE_N))
		=>
		Ignore;
			Wait_Cnf

#	Current State	Event or condition => action	Next state

T4	Wait_Cnf	DLDE-DATA.indication()
		&& DataType == DATA
		&& TakeServiceID(Payload) ==(READ || WRITE || REPORT ACK)
		&& TakeMsgType(Payload) == (RESPONSE_P || RESPONSE_N)
		=>
		VCR_ID:= GetVcrID(SrcAddr, CLIENT, 0);
		StopWatchdogTimer(VCR_ID);
		ASLDE-DATA.confirm(
		    SrcAddr,
		    ServiceID:= TakeServiceID(Payload),
		    AsduLength:= PayloadLength – 4,
		    Asdu:= TakeASLPayload(PayloadLength, Payload)
		)
			Idle

T5	Wait_Cnf	Watchdog Timer with VCR_ID expires
		=>
		ASLDE-DATA.confirm(
		    SrcAddr:= GetPeerAddr(VCR_ID),
		    ServiceID:= RestoreSvrID(VCR_ID),
		    MsgTyep:= RESPONSE_N
		    AsduLength:= 2,
		    Asdu:= BuildErrAsdu(SERVICE_EXPIRATION, 0)
		)
			Idle

10.7.3.8 Server state machine

The server state machine (AMSV) shall have the following states:

– Idle state: the initial and idle state of the server VCR endpoint. In this state, the VCR endpoint is waiting for DLDE-DATA.indication primitive to deliver an ASL packet including a confirmed service request. After receiving the primitive, it shall parse the ASL packet and invoke ASLDE-DATA.indication primitive to deliver the request message to UAP, and then transfer to Wait_Rsp state waiting for the response message.

– Wait_Rsp state: in this state, the server VCR endpoint is waiting for the AL service response message returned from UAP. It shall perform one the the following state transitions:

  • If it receives the service response message through ASLDE-DATA.response primitive within Watchdog interval, it shall pack the response message into an ASL packet and invoke DLDE-DATA.request primitive to send the ASL packet, and then transfer to Idle state.

  • If it does not receive the response message within Watchdog interval, it shall return a negative response with “SERVICE_TIME_EXPIRATION” to the field device, and transfer to Idle state.

Figure 102 and Table 97 show the state transitions of the server state machine.

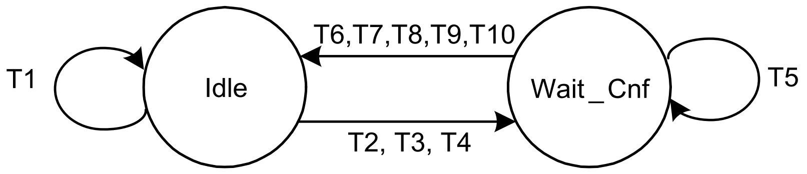

Figure 102 – State transition diagram of AMSV

Table 97 – State transition table of AMSV

#	Current State	Event or condition => action	Next state
T1	Idle	ASLDE-DATA.request() || ASLDE-DATA.response() || DLDE-DATA.indication() && (DataType != DATA || TakeMsgType(Payload) != REQUEST || TakeServiceID(Payload) != (READ || WRITE || REPORT ACK)) => Ignore;	Idle

T2	Idle	DLDE-DATA.indication()
		&& DataType == DATA
		&& TakeServiceID(Payload) == (READ || WRITE || REPORT ACK)
		&& TakeMsgType(Payload) == REQUEST
		&& TakeUAPID(Payload) !=0
		=>
		ServiceID:= TakeServiceID(Payload);
		UAP_ID:= TakeUAPID(Payload);
		VCR_ID:= GetVcrID(UAP_ID, SERVER, 0);
		StartWatchdogTimer(VCR_ID);
		StoreSvrID(VCR_ID, ServiceID);
		ASLDE-DATA.indication(
		    ServiceID:= TakeServiceID(Payload),
		    UAP_ID,
		    AsduLength:= PayloadLength – 4,
		    Asdu:= TakeASLPayload(PayloadLength, Payload)
		)
			Wait_Rsp

#	Current State	Event or condition => action	Next state
T3	Idle	DLDE-DATA.indication() && DataType == DATA && TakeServiceID(Payload) == READ && TakeMsgType(Payload) == REQUEST && TakeUAPID(Payload) ==0 => ServiceID:= TakeServiceID(Payload); VCR_ID:= GetVcrID(0, SERVER, 0); StartWatchdogTimer(VCR_ID); StoreSvrID(VCR_ID, ServiceID); DMAP-MIB-GET.request(Handle, AttributeID, AttributeMemID, FirstStoreIndex, Count)	Wait_Rsp
T4	Idle	DLDE-DATA.indication() && DataType == DATA && TakeServiceID(Payload) == WRITE && TakeMsgType(Payload) == REQUEST && TakeUAPID(Payload) ==0 => ServiceID:= TakeServiceID(Payload); VCR_ID:= GetVcrID(0, SERVER, 0); StartWatchdogTimer(VCR_ID); StoreSvrID(VCR_ID, ServiceID); DMAP-MIB-SET.request(Handle, AttributeID, AttributeMemID, FirstStoreIndex, Count, AttributeValue)	Wait_Rsp
T5	Wait_Rsp	(ASLDE-DATA.response() && ServiceID != (READ || WRITE || REPORT ACK)) || ASLDE-DATA.request() || DLDE-DATA.indication() => Ignore;	Wait_Rsp

#	Current State	Event or condition => action	Next state

T6	Wait_Rsp	ASLDE-DATA.response() && ServiceID == (READ || WRITE || REPORT ACK)
		=>
		VCR_ID:= GetVcrID(UAP_ID, SERVER, 0);
		StopWatchdogTimer(VCR_ID);
		DLDE-DATA.request(
		    VCR_ID
		    DataType:= DATA;
		    Priority:= NRT,
		    PayloadLength:= AsduLength + 4,
		    Payload:= BuildAPDU(ServiceID, MsgType, UAP_ID, AsduLength , Asdu)
		)
			Idle

T7	Wait_Rsp	DMAP-MIB-GET.confirm()
		&& Status ==0
		=>
		VCR:= GetVcrID(0, SERVER, 0);
		StopWatchdogTimer(VCR_ID);
		ServiceID:= READ;
		MsgType:= RESPONSE_P;
		UAP_ID:= 0;
		AsduLength:= Sizeof(AttributeValue);
		Asdu:= AttributeValue;
		DLDE-DATA.request(
		    VCR_ID,
		    DataType:= DATA;
		    Priority:= NRT,
		    PayloadLength:= AsduLength + 4,
		    Payload:= BuildAPDU(ServiceID, MsgType, UAP_ID, AsduLength , Asdu)
		)
			Idle

T8	Wait_Rsp	DMAP-MIB-SET.confirm()
		&& Status ==0
		=>
		VCR:= GetVcrID(0, SERVER, 0);
		StopWatchdogTimer(VCR_ID);
		ServiceID:= WRITE;
		MsgType:= RESPONSE_P;
		UAP_ID:= 0;
		DLDE-DATA.request(
		    VCR_ID,
		    DataType:= DATA;
		    Priority:= NRT,
		    PayloadLength:= 4,
		    Payload:= BuildAPDU(ServiceID, MsgType, UAP_ID, 0 , NULL)
		)
			Idle

#	Current State	Event or condition => action	Next state

T9	Wait_Rsp	(DMAP-MIB-GET.confirm() || DMAP-MIB-SET.confirm())
		&& Status !=0
		=>
		VCR_ID:= GetVcrID(0, SERVER, 0);
		StopWatchdogTimer(VCR_ID);
		ServiceID:= RestoreSvrID(VCR_ID);
		UAP_ID:= 0;
		MsgType:= RESPONSE_N;
		Asdu:= BuildErrAsdu(Status, 0);
		AsduLength:= 2;
		DLDE-DATA.request(
		    DstAddr:= GetPeerAddr(VCR_ID),
		    DataType:= DATA;
		    Priority:= NRT,
		    PayloadLength:= AsduLength + 4,
		    Payload:= BuildAPDU(ServiceID, MsgType, UAP_ID, AsduLength, Asdu)
		)
			Idle

T10	Wait_Rsp	Watchdog Timer with VCR_ID expires
		=>
		ServiceID:= RestoreSvrID(VCR_ID);
		MsgType:= RESPONSE_N;
		Asdu:= BuildErrAsdu(SERVICE_EXPIRATION, 0);
		AsduLength:= 2;
		DLDE-DATA.request(
		    VCR_ID,
		    DataType:= DATA,
		    Priority:= NRT,
		    PayloadLength:= AsduLength + 4,
		    Payload:= BuildAPDU(ServiceID, MsgType, 0, AsduLength, Asdu)
		)
			Idle

10.7.3.9 Publisher state machine

The publisher state machine (AMPB) shall have the following states:

– Init state: the initial state of the publisher VCR endpoint. After completing configuration, the VCR endpoint shall perform one the the following state transitions:
  - If the value of VCRActiveTime is 0, it shall enter Active state; or
  - If the value of VCRActiveTime is non-zero, it shall enter NO_Active state;

– No_Active: in this state, the publisher VCR endpoint has already been configured but not activated. If VCRActiveTime expires, it shall enter Active state;

– Active: the activative state of the publisher VCR endpoint, in which it is waiting for ASLDE-DATA.request primitive to deliver a PUBLISH request message. The VCR endpoint shall perform one the the following state transitions:

- If receiving PUBLISH request message, it shall put the message into its buffer, and pack the message into a ASL packet, invoke DLDE-DATA.request primitive to send the ASL packet, and keep in Active state; or
- If DataUpdateRate expires, it shall get the data from its buffer to form a ASL packet, invoke DLDE-DATA.request primitive to send the ASL packet, and keep in Active state;

Figure 103 and Table 98 show the state transitions of the publisher state machine.

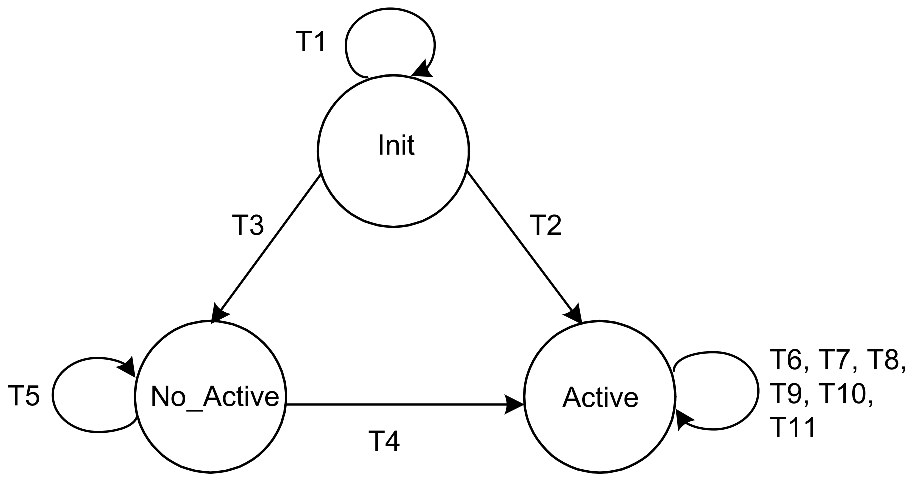

Figure 103 – State transition diagram of AMPB

Table 98 – State transition table of AMPB

#	Current State	Event or condition => action	Next State

T1	Init	ASLDE-DATA.request() || ASLDE-DATA.response() || DLDE-DATA.indication()
		=>
		Ignore;
			Init

T2	Init	VCR with VCR_ID configured completely
		&& VCRActiveTime == 0
		=>
		CreatBuffer(VCR_ID);
		StartDataUpdateRateTimer(VCR_ID);
		StartDeadlineTimer(VCR_ID);
			Active

T3	Init	VCR with VCR_ID configured completely && VCRActiveTime != 0 => CreatBuffer(VCR_ID); StartActiveTimer(VCR_ID);	No_Active
T4	No_Active	VCRActiveTime with VCR_ID expires => StartDataUpdateRateTimer(VCR_ID);	Active

T5	No_Active	ASLDE-DATA.request() || ASLDE-DATA.response() || DLDE-DATA.indication()
		=>
		Ignore;
			No_Active

#	Current State	Event or condition => action	Next State
T6	Active	(ASLDE-DATA.request() && ServiceID != PUBLISH) || ASLDE-DATA.response() || DLDE-DATA.indication() => Ignore;	Active

T7	Active	(ASLDE-DATA.request() && ServiceID == PUBLISH)
		&& CheckEvent(TakeUAPID(Payload), PROCESS_DATA_NOT_UPDATED)
		=>
		VCR_ID:= GetVcrID(DstAddr, PUBLISHER, UAP_ID);
		PutDataIntoBuffer(VCR_ID, AsduLength, Asdu);
		UAP_ID:= GetUAPID(VCR_ID);
		ServiceID:= PUBLISH;
		MsgType:= REQUEST;
		DLDE-DATA.request(
		    VCR_ID,
		    DataType:= DATA,
		    Priority:= RT1,
		    PayloadLength:= AsduLength + 4,
		    Payload:= BuildAPDU(ServiceID, MsgType, UAP_ID, AsduLength, Asdu)
		)
		StartDataUpdateRateTimer(VCR_ID);
		StartDeadlineTimer(VCR_ID);
		SetEvent(UAP_ID, PROCESS_DATA_NOT_UPDATED, DISAPPEAR);
			Active

T8	Active	(ASLDE-DATA.request() && ServiceID == PUBLISH)
		&&! CheckEvent(TakeUAPID(Payload), PROCESS_DATA_NOT_UPDATED)
		=>
		VCR_ID:= GetVcrID(DstAddr, PUBLISHER, UAP_ID);
		PutDataIntoBuffer(VCR_ID, AsduLength, Asdu);
		UAP_ID:= GetUAPID(VCR_ID);
		ServiceID:= PUBLISH;
		MsgType:= REQUEST;
		DLDE-DATA.request(
		    VCR_ID,
		    DataType:= DATA,
		    Priority:= RT1,
		    PayloadLength:= AsduLength + 4,
		    Payload:= BuildAPDU(ServiceID, MsgType, UAP_ID, AsduLength, Asdu)
		)
		StartDataUpdateRateTimer(VCR_ID);
		StartDeadlineTimer(VCR_ID);
			Active

#	Current State	Event or condition => action	Next State
T9	Active	DataUpdateRate timer with VCR_ID expires => UAP_ID:= GetUAPID(VCR_ID); ServiceID:= PUBLISH; MsgType:= REQUEST; Asdu:= GetDataFromBuffer(VCR_ID); AsduLength:= Sizeof(Asdu); DLDE-DATA.request(VCR_ID, DataType:= DATA, Priority:= RT1, PayloadLength:= AsduLength + 4, Payload:= BuildAPDU(ServiceID, MsgType, UAP_ID, AsduLength, Asdu)) StartDataUpdateRateTimer(VCR_ID)	Active
T10	Active	Deadline timer with VCR_ID expires && !CheckEvent(TakeUAPID(Payload), PROCESS_DATA_NOT_UPDATED) => UAP_ID:= GetUAPID(VCR_ID); SetEvent(UAP_ID, PROCESS_DATA_NOT_UPDATED, APPEAR); StartDeadlineTimer(VCR_ID)	Active
T11	Active	Deadline timer with VCR_ID expires && CheckEvent(TakeUAPID(Payload), PROCESS_DATA_NOT_UPDATED) => StartDeadlineTimer(VCR_ID)	Active

10.7.3.10 Subscriber state machine

The subscriber state machine (AMSB) shall have the following states:

– Init state: the initial state of the subscriber VCR endpoint. After completing configuration, the VCR endpoint shall perform one of the following state transitions:
  - If the value of VCRActiveTime is 0, it shall enter Active state; or
  - If the value of VCRActiveTime is non-zero, it shall enter NO_Active state;

– No_Active: in this state, the subscriber VCR endpoint has already been configured but not activated. If VCRActiveTime expires, it shall enter Active state;

– Active: the activative state of the subscriber VCR endpoint, in which it is waiting for DLDE-DATA.indication primitive to deliver an ASL packet including a PUBLISH request. The VCR endpoint shall perform one the the following state transitions:
  - If receiving a PUBLISH request, it shall put the request message into its buffer and invoke ASLDE-DATA.indication primitive to deliver the request to UAP. The VCR endpoint shall also check the EventData's "PROCESS DATA NOT UPDATED IN TIME" EventFlag bit of the UAOs that belong to the UAP. If the bit value is 1, it shall be set to 0 in order to generate an alarm event of "PROCESS DATA NOT UPDATED IN TIME" disappearance; or
  - If DataUpdateRate expires, it shall get the data from its buffer, invoke ASLDE-DATA.indication primitive to send the data, and keep in Active state. In this case, the

data is the PUBLISH request message that has been stored in the buffer for the last time; or

- If Deadline time expires, it shall check the EventData's "PROCESS DATA NOT UPDATED IN TIME" EventFlag bit of the UAOs that belong to the UAP. If the bit value is 0, it shall be set to 1 in order to generate an alarm event of "PROCESS DATA NOT UPDATED IN TIME" occurrence. And it shall keep in Active state.

Figure 104 and Table 99 show the state transitions of the subscriber state machine.

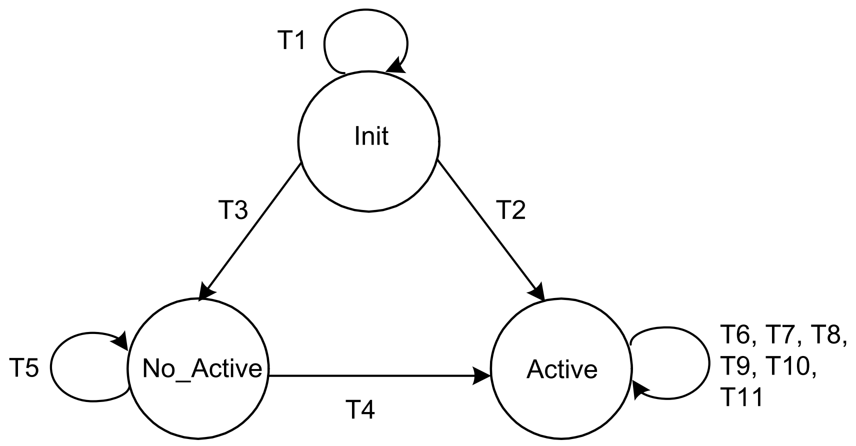

Figure 104 – State transition diagram of AMSB

Table 99 – State transition table of AMSB

#	Current State	Event or condition => action	Next State

T1	Init	ASLDE-DATA.request() || ASLDE-DATA.response() || DLDE-DATA.indication()
		=>
		Ignore;
			Init

T2	Init	VCR with VCR_ID configured completely
		&& VCRActiveTime == 0
		=>
		CreatBuffer(VCR_ID);
		StartDataUpdateRateTimer(VCR_ID);
		StartDeadlineTimer(VCR_ID);
			Active

T3	Init	VCR with VCR_ID configured completely && VCRActiveTime != 0 => CreatBuffer(VCR_ID); StartActiveTimer(VCR_ID);	No_Active

T4	No_Active	VCRActiveTime with VCR_ID expires
		=>
		StartDataUpdateRateTimer(VCR_ID)
		StartDeadlineTimer(VCR_ID);
			Active

T5	No_Active	ASLDE-DATA.request() || ASLDE-DATA.response() || DLDE-DATA.indication()
		=>
		Ignore;
			No_Active

#	Current State	Event or condition => action	Next State

T6	Active	ASLDE-DATA.request() || ASLDE-DATA.response()
		|| (DLDE-DATA.indication()
		  && (TakeServiceID(Payload) != PUBLISH || TakeMsgType(Payload) != REQUEST
		      || DataType != DATA)
		=>
		Ignore;
			Active

T7	Active	DLDE-DATA.indication()
		&& DataType == DATA;
		&& TakeServiceID(Payload) == PUBLISH
		&& TakeMsgType(Payload) == REQUEST
		&& GetUAPID(GetVcrID(SrcAddr, SUBSCRIBER, TakeUAPID(Payload))) == TakeUAPID(Payload)
		&& CheckEvent(TakeUAPID(Payload), PROCESS_DATA_NOT_UPDATED)
		=>
		UAP_ID:= TakeUAPID(Payload);
		VCR_ID:= GetVcrID(SrcAddr, SUBSCRIBER, UAP_ID);
		ServiceID:= TakeServiceID(Payload);
		AsduLength:= PayloadLength – 4;
		Asdu:= TakeASLPayload(PayloadLength, Payload);
		PutDataIntoBuffer(VCR_ID, AsduLength, Asdu);
		ASLDE-DATA.indication(
		    ServiceID,
		    UAP_ID,
		    AsduLength,
		    Asdu
		)
		StartDataUpdateRateTimer(VCR_ID);
		StartDeadlineTimer(VCR_ID);
		SetEvent(UAP_ID, PROCESS_DATA_NOT_UPDATED, DISAPPEAR);
			Active

#	Current State	Event or condition => action	Next State

T8	Active	DLDE-DATA.indication()
		&& DataType == DATA
		&& TakeServiceID(Payload) == PUBLISH
		&& TakeMsgType(Payload) == REQUEST
		&& GetUAPID(GetVcrID(SrcAddr, SUBSCRIBER, TakeUAPID(Payload))) == TakeUAPID(Payload)
		&& !CheckEvent(TakeUAPID(Payload), PROCESS_DATA_NOT_UPDATED)
		=>
		UAP_ID:= TakeUAPID(Payload);
		VCR_ID:= GetVcrID(SrcAddr, SUBSCRIBER, UAP_ID);
		ServiceID:= TakeServiceID(Payload);
		AsduLength:= PayloadLength – 4;
		Asdu:= TakeASLPayload(PayloadLength, Payload);
		PutDataIntoBuffer(VCR_ID, AsduLength, Asdu);
		ASLDE-DATA.indication(
		    ServiceID,
		    UAP_ID,
		    AsduLength;
		    Asdu;
		)
		StartDataUpdateRateTimer(VCR_ID);
		StartDeadlineTimer(VCR_ID);
			Active

T9	Active	DatyUpdateCycle Timer with VCR_ID expires
		=>
		Asdu:= GetDataFromBuffer(VCR_ID);
		AsduLength:= Sizeof(Asdu);
		UAP_ID:= GetUAPID(VCR_ID);
		ASLDE-DATA.indication(
		    ServiceID:= PUBLISH,
		    UAP_ID,
		    AsduLength;
		    Asdu;
		)
		StartDataUpdateRateTimer(VCR_ID);
			Active

T10	Active	Deadline timer with VCR_ID expires
		&& !CheckEvent(TakeUAPID(Payload), PROCESS_DATA_NOT_UPDATED)
		=>
		UAP_ID:= GetUAPID(VCR_ID);
		SetEvent(UAP_ID, PROCESS_DATA_NOT_UPDATED, APPEAR);
		StartDeadlineTimer(VCR_ID)
			Active

T11	Active	Deadline timer with VCR_ID expires && CheckEvent(TakeUAPID(Payload), PROCESS_DATA_NOT_UPDATED) => StartDeadlineTimer(VCR_ID)	Active

10.7.3.11 Report source state machine

The report source state machine (AMRS) shall have the following states:

– Active = the report source VCR endpoint only have Active state.

Figure 105 and Table 100 show the state transitions of the report source state machine.

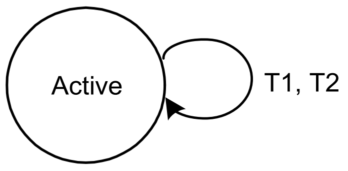

Figure 105 – State transition diagram of AMRS

Table 100 – State transition table of AMRS

#	Current State	Event or condition => action	Next State
T1	Active	ASLDE-DATA.response() || (ASLDE-DATA.request() && ServiceID != REPORT) || DLDE-DATA.indication() => Ignore;	Active

T2	Active	ASLDE-DATA.request() && ServiceID == REPORT
		=>
		VCR_ID:= GetVcrID(DstAddr, ServiceID, UAP_ID)
		Msgtype:= REQUEST;
		DLDE-DATA.request(
		    VCR_ID,
		    DataType:= DATA,
		    Priority:= Priority,
		    PayloadLength:= AsduLength + 4,
		    Payload:= BuildAPDU(ServiceID, Msgtype, UAP_ID, AsduLength, Asdu)
		)
			Active

10.7.3.12 Report sink state machine

The report sink state machine (AMRK) shall have the following states:

– Active = the report sink VCR endpoint only have Active state.

Figure 106 and Table 101 show the state transitions of the report sink state machine.

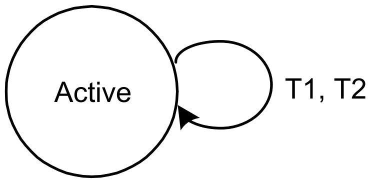

Figure 106 – State transition diagram of AMRK

Table 101 – State transition table of AMRK

#	Current State	Event or condition => action	Next State
T1	Active	ASLDE-DATA.request() || ASLDE-DATA.response() || (DLDE-DATA.indication() && (DataType != DATA || TakeServiceID(Payload) != REPORT || TakeMsgType(Payload) != REQUEST) => Ignore	Active

T2	Active	DLDE-DATA.indication()
		&& DataType == DATA
		&& TakeServiceID(Payload) == REPORT
		&& TakeMsgType(Payload) != REQUEST
		=>
		UAP_ID:= TakeUapID(Payload);
		VCR_ID:= GetVcrID(SrcAddr, REPORT_SINK, UAP_ID);
		ASLDE-DATA.indication(
		ServiceID:= TakeServiceID(Payload),
		UAP_ID:= TakeUAPID(Payload),
		AsduLength:= PayloadLength – 4,
		Asdu:= TakeASLPayload(PayloadLength, Payload)
		)
			Active

10.7.3.13 Functions used in the ASL state machines

All Functions used in the ASL state machines are shown in Table 102.

Table 102 – All Functions in ASLM

Function	Input	Output	Description
BuildAPDU	ServiceID Msgtype UAP_ID AsduLength Asdu	Apdu	Adding packet header to the ASDU according to ASL general packet format and build an APDU
BuildErrAsdu	ErrorCode AddInfo	Asdu	Building an AL negative service response message
CheckEvent	UAP_ID EventFlag	True | False	Checking whether the EventFlag bit of EventData of the the UAOs belonging to the UAP is set.
CreatBuffer	VCR_ID		Creating a data buffer of the VCR endpoint to save the input/output data
GetDataFromBuffer	VCR_ID	Asdu	Getting data from the buffer of the VCR endpoint
GetPeerAddr	VCR_ID	PeerAddr	Getting PeerAddr value of the VCR endpoint
GetUAPID	VCR_ID	UAP_ID	Getting UAP_ID value of the VCR endpoint
GetVcrID	DstAddr VcrEpType UAP_ID	VCR_ID	Getting VCR_ID value according the Address, VcrEpType, and UAP_ID
PutDataIntoBuffer	VCR_ID AsduLength Asdu		Putting data into the buffer of the VCR endpoint
RestoreSvrID	VCR_ID	ServiceID	Obtaining ServiceID value saved previously at the VCR endpoint
SetEvent	UAP_ID EventFlag AppearFlag		Setting the EventFlag bit of EventData of the UAOs belonging to the UAP with the value of AppearFlag
Sizeof	Data	Length	Calculating the octet length of the data
StartActiveTimer	VCR_ID		Starting the VCRActiveTime timer of the VCR endpoint
StartDataUpdateRateTime	VCR_ID		Starting the DataUpdateRate timer of the VCR endpoint
StartDeadlineTimer	VCR_ID		Starting the Deadline timer of the VCR endpoint
StartWatchdogTimer	VCR_ID		Start the Watchdog timer of the VCR endpoint
StoreSvrID	VCR_ID ServiceID		Storing ServiceID value at the VCR endpoint
TakeASLPayload	DllPayloadLength DllPayload	Asdu	Obtaining ASDU from DLL payload according ASL general packet format
TakeMsgType	DllPayload	Msgtype	Obtaining Msgtype value from DLL payload according ASL general packet format
TakeServiceID	DllPayload	ServiceID	Obtaining ServiceID value from DLL payload according ASL general packet format
TakeUAPID	DllPayload	UAP_ID	Obtaining UAP_ID value from DLL payload according ASL general packet format

11 Security

11.1 General

11.1.1 Security management architecture

Considering the security management requirements in real-time and resource constrained factory automation network, multiple security measures are needed to construct the security management architecture for WIA-FA network.

As shown in Figure 107, the security management architecture of WIA-FA network consists of security manager (SM) in the gateway device and security management modules in each access device and field device.

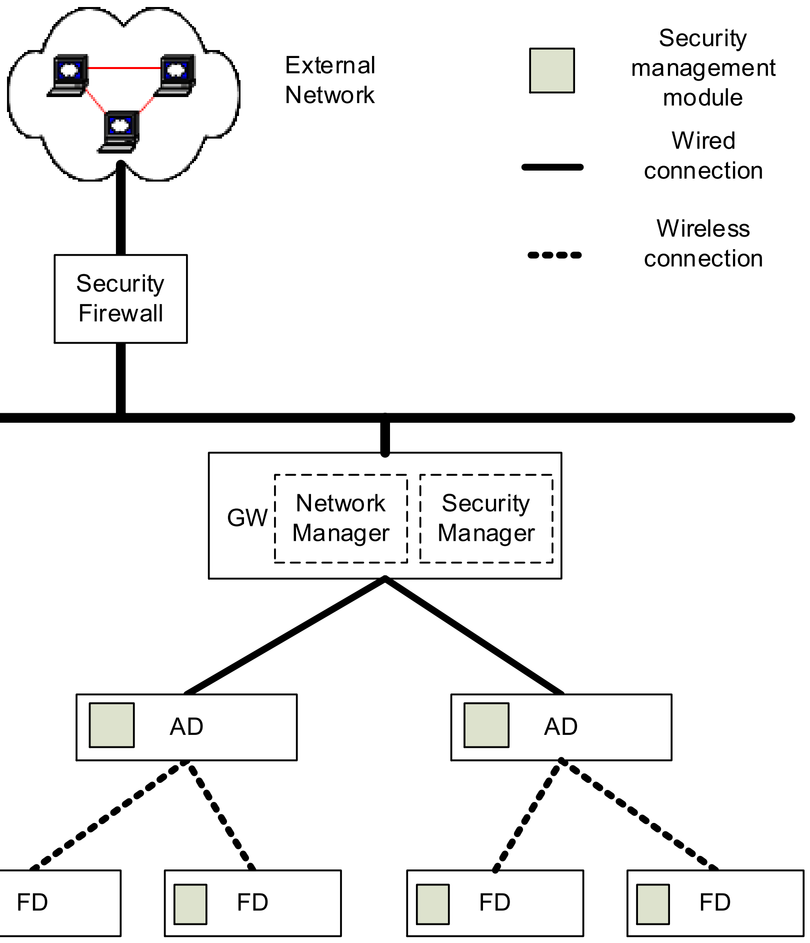

Figure 107 – Security management architecture

As shown in Figure 107, DMAP in gateway device includes a SM. SM shall perform the following functions:

- Configuring the system security policy (see Clause A.4) of a WIA-FA network by setting SecLevel attribute of gateway device, access device and field device according to specific applications.
- Authenticating field device attempting to join WIA-FA network.
- Managing keys of the whole network, including key generating, key establishment and key updating.
- Handling security alarm report from field device and access device.

Access device includes a security management module with the following functions:

– Managing the keys for secure DLL communication;
– Implementing security alarm mechanism.

The implementations of the security functions of access device are described in Clause 9.

DMAP of field device includes a security management module with the following functions:

– Conducting secure join process of field device;
– Managing the keys for secure joining, secure DLL communication and secure key distribution;
– Implementing security alarm mechanism.

The security management architecture interacts with external networks through a firewall, which guarantees the normal operations of the WIA-FA network. The firewall protects the border of the whole WIA-FA network, which is not specified in this PAS.

11.1.2 Security functions

WIA-FA network provides the following security functions:

– Device authentication;
– Data integrity;
– Data confidentiality;
– Replay attack protection;
– Key management;
– Security alarm.

Device authentication function is accomplished by join service. Data integrity, data confidentiality and replay attack protection are accomplished by DLL data service. Key management function is accomplished by key establish service and key update service. Security alarm function is accomplished by security alarm service.

11.1.3 Keys

The SM generates the keys used in the WIA-FA network. Key is 128-bit information described by Key_Struct (see Table 20) and stored in each device's MIB.

The following cryptographic keys are defined, and each key is responsible for a dedicated security function.

– Shared Key (KS): KS is distributed in the provision period by the SM through handheld device. It is used when no data encryption key has been distributed and is shared by the whole WIA-FA network.
– Join Key (KJ): KJ is distributed in the provision period by the SM through handheld device. It is used for authenticating a new device attempting to join the WIA-FA network.
– Key Encryption Key (KEK): KEK is distributed by the SM after a device has joined the WIA-FA network. It is used to protect keys during the keys' updating.
– Unicast Data Encryption Key (KEDU): KEDU is distributed by the SM after a device has joined the WIA-FA network. It is used to protect the confidentiality and integrity of the unicasted frames between access device and field device, and each field device's data in aggregated frames. Access devices in the same set share the same KEDU with the same field device.
– Broadcast Data Encryption Key (KEDB): KEDB is distributed by the SM after a device has joined the WIA-FA network. It is used to protect the confidentiality and integrity when access device broadcasting frames to field devices.

In order to protect the key, keys have different life cycle, as shown in Figure 108.

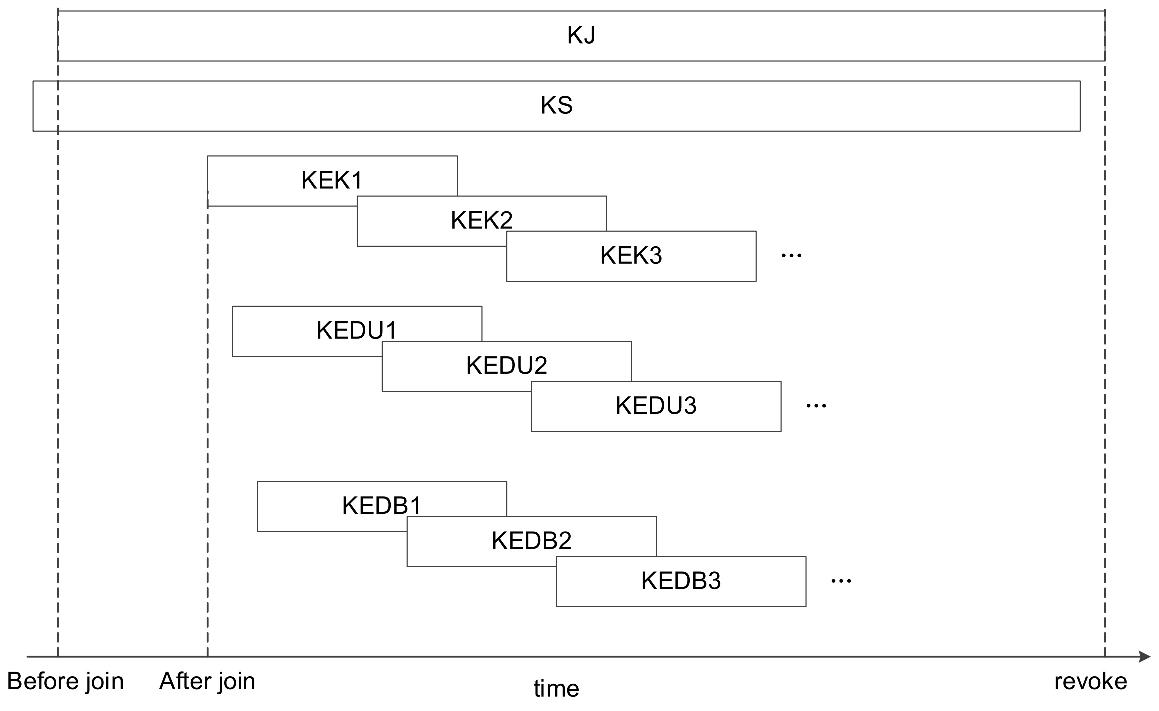

Figure 108 – Life cycle of keys

From establishment to abolition, a key experiences the following states:

– BACKUP: a key is in this state when it has been distributed and stored in MIB but its KeyActiveSlot (see 6.7.1.2.1) has not achieved;

– USING: a key is in this state when KeyActiveSlot of the key is achieved, and the key is being used;

– EXPIRED: a key is in this state when KeyActiveSlot+KeyUpdateDur (see 6.7.1.2.1) of the key is achieved, but the key for its updating is unavailable. In this status the key is still used;

– INVALID: a key is in this state when the key for its updating is in use.

After a field device has joined the network with SecLevel not equal to 0/1, at one time, each kind of key has only one available key in USING or EXPIRED state at most.

11.2 Security services

11.2.1 General

Security Services include key establish service, key update service and security alarm service. They are provided by the security management module of field devices and access devices together with security manager in GW, and accessed through DLME-SAP.

The key establish service, key update service and security alarm service for access devices are defined in Clause 9. This section only defines the key establish service, key update service and security alarm service for field devices.

11.2.2 Key establish service

11.2.2.1 Key establish request primitive

The key establish request primitive KEY-ESTABLISH.request allows the security manager in GW to request DLL to send key establish request frame.

```
KEY-ESTABLISH.request(
                                DstAddr,
                                KeyMaterial
                                )
```

Table 103 specifies the parameters for KEY-ESTABLISH.request.

Table 103 – Parameters for KEY-ESTABLISH.request

Name	Data type	Valid range	Description
DstAddr	Unsigned16	0 to 65 535	Short address of destination device
KeyMaterial	KeyMaterial_Struct		Key related information and its message integrity code, see Table 104 for its format

The key material format for distribution is shown in Table 104.

Table 104 – KeyMaterial_Struct structure

Name	Data type	Valid range	Description
KeyID	Unsigned16	0 to 65 535	Identifier of the key
KeyType	Unsigned8	0 to 255	Type of the key, see Table 20
KeyActiveSlot	Unsigned48	0 to (2^{48}-1)	Absolute timeslot number to activate the key
KeyDataValue	KeyData	-	Key data, encrypted in CCM* with KJ or KEK
KeyMIC	Unsigned32	0 to (2^{32}-1)	Message integrity code, generated by CCM* with KJ or KEK, the protecting range involves: KeyID, KeyType, KeyActiveSlot and KeyDataValue.

Since the key related information should be protected in communication, the security manager in gateway device should protect it in CCM*. KJ of the destination field device should be used while the key establish service is conducted, and KEK of the destination field device should be used while the key update service is conducted.

The 13-otect NONCE needed in CCM* is defined in Figure 109:

PhyAddress	TimeStamp	SecurityLevel
8 octets	4 octets	1 octet

Figure 109 – Format of NONCE

Where the PhyAddress is the 64 bits physical address of the field device; the TimeStamp is the low 4 bytes of the KeyActiveSlot; and the SecurityLevel is set to 5, i.e. Encryption & MIC-32.

11.2.2.2 Key establish indication primitive

Key establish indication primitive KEY-ESTABLISH.indication is used for the DLL of field device to indicate the reception of key establish request frame, and is issued to security management module of field device.

```
KEY-ESTABLISH.indication(
                              KeyMaterial
                              )
```

Table 105 specifies the parameters for KEY-ESTABLISH.indication.

Table 105 – Parameters for KEY-ESTABLISH.indication

Name	Data type	Valid range	Description
KeyMaterial	KeyMaterial_Struct		See Table 104

Upon receipt of this primitive, the security management module in field device decrypts the KeyDataValue with KJ, and checks KeyMIC, and, if KeyMIC is correct, generates a related Key_Struct record for this key and stores it in KeyList.

11.2.2.3 Key establish response primitive

Key establish response primitive KEY-ESTABLISH.response is generated by the security management module of field device, and issued to DLL to send key establish response frame.

```
KEY-ESTABLISH.response(
                                   KeyID,
                                   Status
                                   )
```

Table 106 specifies the parameters for KEY-ESTABLISH.response.

Table 106 – Parameters for KEY-ESTABLISH.response

Name	Data type	Valid range	Description
KeyID	Unsigned16	0 to 65 535	Identifier of the key

Status	Unsigned8	0 to 255	Execution result of the key establish request:
			0 = SUCCESS;
			1 = FAILURE;
			Others are reserved.

11.2.2.4 Key establish confirm primitive

Key establish confirm primitive KEY-ESTABLISH.confirm is used for the DLL of access device to inform the security manager in gateway device whether the key establishment is success.

```
KEY-ESTABLISH.confirm(
                              DstAddr,
                              KeyID,
                              Status
                              )
```

Table 107 specifies the parameters for KEY-ESTABLISH.confirm.

Table 107 – Parameters for KEY-ESTABLISH.confirm

Name	Data type	Valid range	Description
DstAddr	Unsigned16	0 to 65 535	Short address of field device
KeyID	Unsigned16	0 to 65 535	Identifier of the key

Status	Unsigned8	0 to 255	Execution result of the key establish request:
			0 = SUCCESS;
			1 = FAILURE;
			Others are reserved.

11.2.2.5 Time sequence of key establishment

Figure 110 shows the time sequence of key establishment.

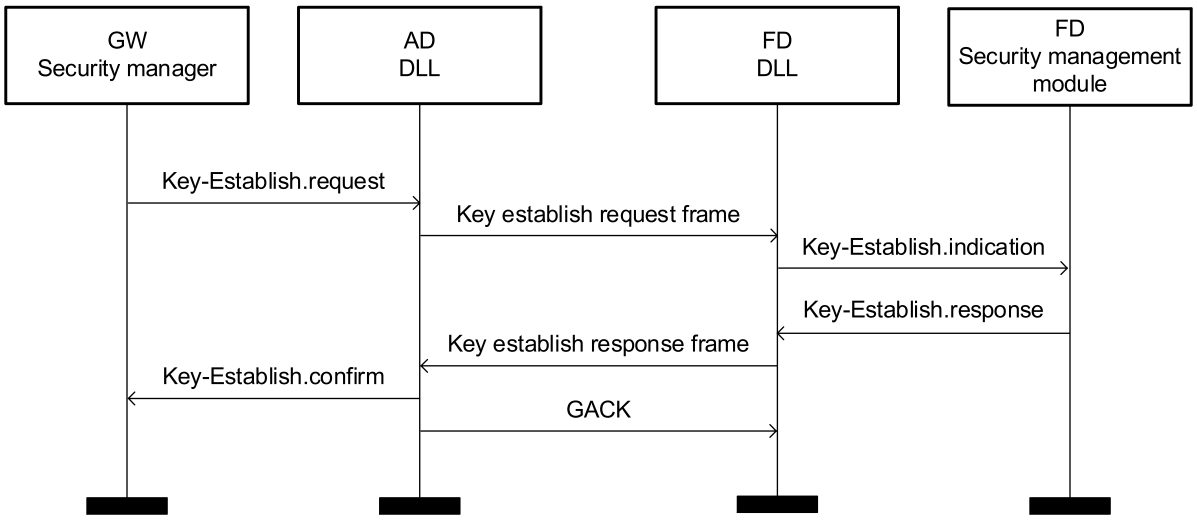

Figure 110 – Time sequence of key establishment

11.2.3 Key update service

11.2.3.1 Key update request primitive

The key update request primitive KEY-UPDATE.request allows the security manager in GW to request DLL to send key update request frame.

```
KEY-UPDATE.request(
                    DstAddr,
                    KeyMaterial
                    )
```

Table 108 specifies the parameters for KEY-UPDATE.request.

Table 108 – Parameters for KEY-UPDATE.request

Name	Data type	Valid range	Description
DstAddr	Unsigned16	0 to 65 535	Short address of field device
KeyMaterial	KeyMaterial_Struct		See Table 104

11.2.3.2 Key update indication primitive

Key update indication primitive KEY-UPDATE.indication is used for the DLL of field device to indicate the reception of key update request frame, and is issued to security management module of field device.

```
KEY-UPDATE.indication(
                              KeyMaterial
                              )
```

Table 109 specifies the parameters for KEY-UPDATE.indication.

Table 109 – Parameters for KEY-UPDATE.indication

Name	Data type	Valid range	Description
KeyMaterial	KeyMaterial_Struct		See Table 104

Upon receipt of this primitive, the security management module in field device decrypts the KeyDataValue with KEK, and KeyMIC, and if KeyMIC is correct, generates a related Key_Struct record for this key and stores it in KeyList.

11.2.3.3 Key update response primitive

Key update response primitive KEY-UPDATE.response is generated by the security management module of field device, and issued to DLL to send key update response frame.

```
KEY-UPDATE.response(
                              KeyID,
                              Status
                              )
```

Table 110 specifies the parameters for KEY-UPDATE.response.

Table 110 – Parameters for KEY-UPDATE.response

Name	Data type	Valid range	Description
KeyID	Unsigned16	0 to 65 535	Identifier of the key

Status	Unsigned8	0 to 255	Execution result of the key update request:
			0 = SUCCESS;
			1 = FAILURE;
			Others are reserved.

11.2.3.4 Key update confirm primitive

Key update confirm primitive KEY-UPDATE.confirm is used for the DLL of access device to inform the security manager in gateway device whether the key update is success.

```
KEY-UPDATE.confirm(
                              DstAddr,
                              KeyID,
                              Status
                              )
```

Table 111 specifies the parameters for KEY-UPDATE.confirm.

Table 111 – Parameters for KEY-UPDATE.confirm

Name	Data type	Valid range	Description
DstAddr	Unsigned16	0 to 65 535	Short address of field device
KeyID	Unsigned16	0 to 65 535	Identifier of the key

Status	Unsigned8	0 to 255	Execution result of the key update request:
			0 = SUCCESS;
			1 = FAILURE;
			Others are reserved.

11.2.3.5 Time sequence of key update

Figure 111 shows the time sequence of key update.

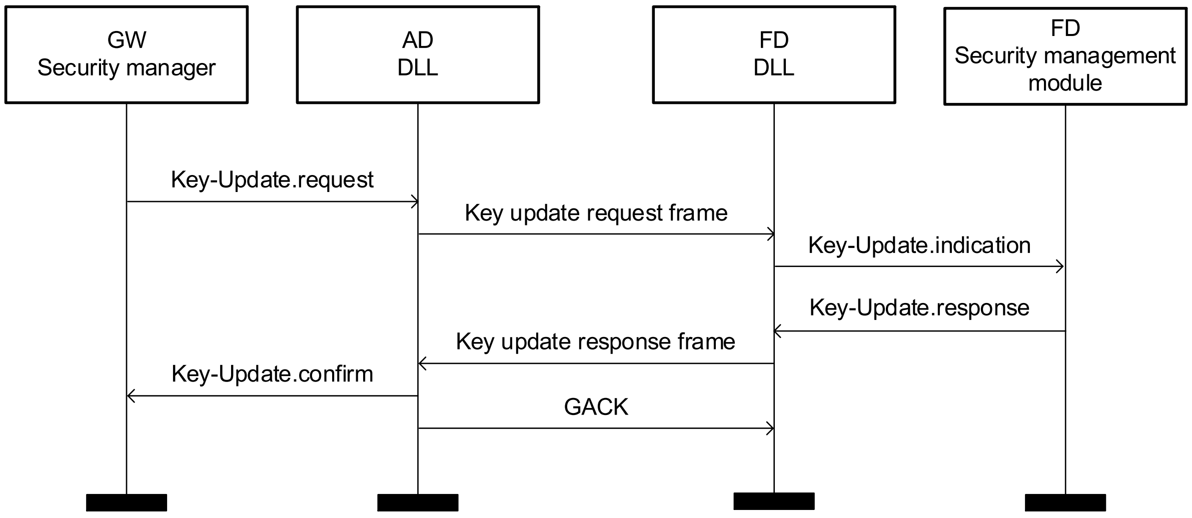

Figure 111 – Time sequence of key updating

11.2.4 Security alarm service

11.2.4.1 Security alarm request primitive

The security alarm request primitive SEC-ALARM.request allows the security management module in FD to request DLL to send security alarm request frame.

```
SEC-ALARM.request(
                  SecAlarmCount,
                  SecAlarmList
                  )
```

Table 112 specifies the parameters for SEC-ALARM.request.

Table 112 – Parameters for SEC-ALARM.request

Name	Data type	Valid range	Description
SecAlarmCount	Unsigned8	0 to 255	The count of security alarms
SecAlarmList	List of SecAlarm_Struct		Security Alarm list, see Figure 112 for its format

The structure of SecAlarm_Struct is shown in Figure 112.

KeyID	AlarmFlag
2 octets	1 octet

Figure 112 – SecAlarmt_Struct structure

The SecAlarmList contains n alarms and n is specified by the SecAlarmCount. Each alarm in the list contains two fields:

– KeyID: the identifier of the key related to the alarm;

– AlarmFlag: is equal to the AlarmFlag (see Table 20) of the key related to the alarm.

11.2.4.2 Security alarm indication primitive

Security alarm indication primitive SEC-ALARM.indication is used for the DLL of access device to indicate the reception of security alarm request frame, and is issued to security manager of gateway device.

```
SEC-ALARM.indication(
                    SrcAddr,
                    SecAlarmCount,
                    SecAlarmList
                    )
```

Table 113 specifies the parameters for SEC-ALARM.indication.

Table 113 – Parameters for SEC-ALARM.indication

Name	Data type	Valid range	Description
SrcAddr	Unsigned16	0 to 65 535	The short address of the field device sending alarm
SecAlarmCount	Unsigned8	0 to 255	The count of security alarms
SecAlarmList	List of SecAlarm_Struct		Security Alarm list, see Figure 112 for its format

Upon receipt of this primitive, the security manager shall update the corresponding keys related to the alarms in the SecAlarmList.

11.2.4.3 Time sequence of security alarm

Figure 113 shows the time sequence of security alarm.

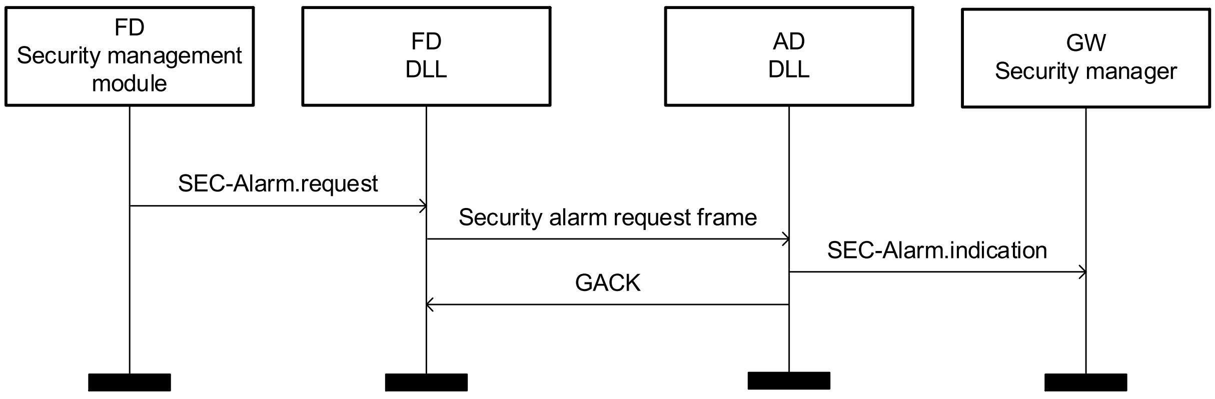

Figure 113 – Time sequence of security alarm

11.3 Secure join

11.3.1 General

When the SecLevel of field device does not equal to 0, field device shall be authenticated by security manager during join process. The authentication is performed with the physical address and the KJ of field device.

For secure join, the filed device and the network manager in gateway device utilize the join primitives, i.e. DLME-JOIN.request and DLME-JOIN.indication (see 8.3.4), with SecMaterial calculated as follows.

SecMaterial = low 64 bits of HMAC(KJ, PhyAddress)

Where, HMAC is based on MD5. See Handbook of Applied Crytography.

11.3.2 Secure join process of FD

FD should be provision the following information before secure joining WIA-FA network:

- Network ID;
- SecLevel;
- KJ and KS.

The secure join process after provision period is as follows:

a) Access devices broadcast beacons periodically;

A field device attempting to join the WIA-FA network continually scans the available channels to get the beacons from access devices and synchronizes with gateway device by using one-way time synchronization method (see 8.1.4);

The field device selects a channel, on which the field device receives beacon frame, gets the shared timeslot and channel information for join request, and sends the join request with security material, i.e. SecMaterial defined in Table 104 in the shared timeslot. The shared timeslot is determined by the "First shared timeslot number" and the “Shared timeslot count” (see 8.4.6) in the beacon frame;

After receiving the join request, network manager in the gateway device forwards the security material and the physical address of the field device to security manager;

Security manager authenticates the field device using the security material and the physical address, and the KJ of the field device, if success, replies authentication success to network manager, otherwise, replies authentication failure to network manager;

Network manager returns the join response to the field device, in which Status is set to 2, if authentication failure, see 8.3.4.3 for the definition of Status;

Field device receives the join response. If Status of join response is 2, field device will repeat the join process again.

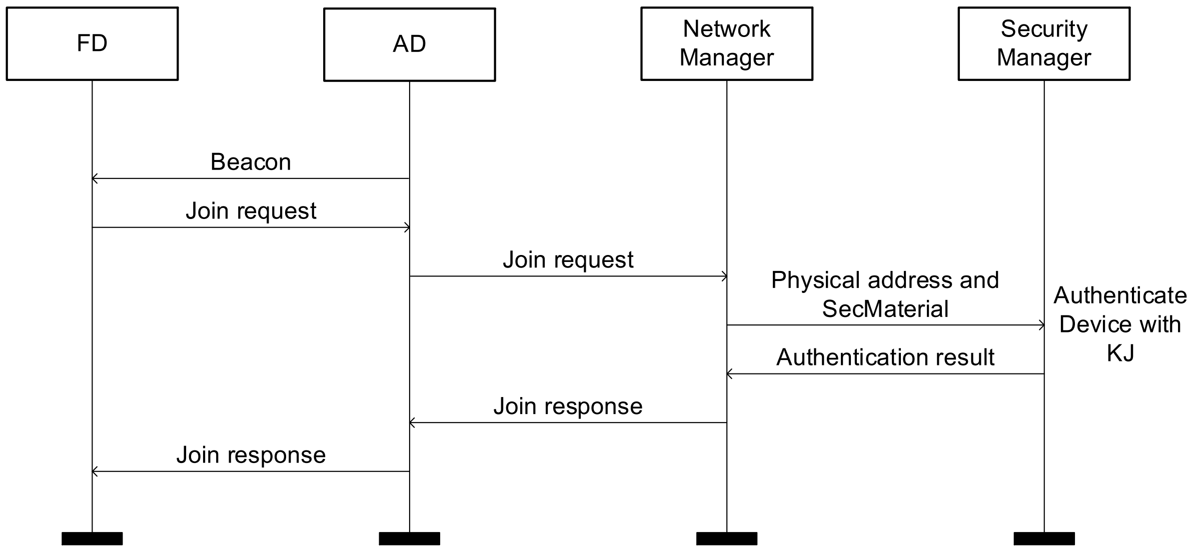

Figure 114 – Secure join process of field device

11.4 Key management

11.4.1 General

Key management includes key establish and key update. Key establish is used for the security manager to establish new keys for the field device; Key update is used for the security manager to update the existing keys in the field device according to its life cycle.

11.4.2 Key establish process

After a field device has joined the network, security manager shall establish KEK, KEDU and KEDB for the field device, as shown in Figure 115.

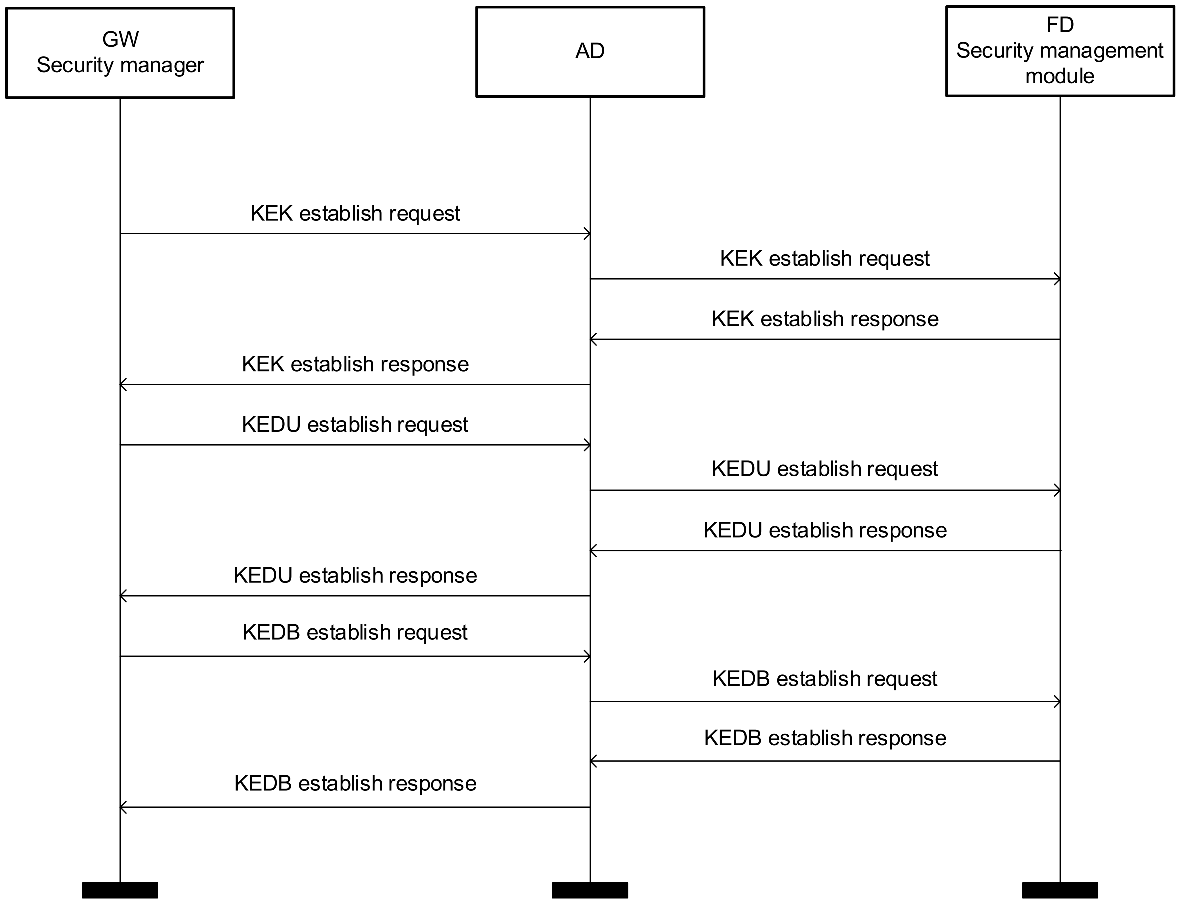

Figure 115 – Key establish process for field device

11.4.3 Key update process

The security manager should update all the KEK, KEDU and KEDB being used in the network before the KeyActiveSlot+KeyUpdateDur of each key.

For each KEK, KEDU and KEDB in the field device, the key being used is defined as Current_key, and the key to be used is defined as New_key.

The defined states and their description of key update are shown in Table 114.

Figure 116 and Table 115 show the state machine and the state transition of key update.

Table 114 – Key update states

State Name	Description
ST1	Current_key: NON, New_key: BACKUP
ST2	Current_key: USING, New_key: NON
ST3	Current_key: USING, New_key: BACKUP
ST4	Current_key: EXPIRED, New_key: BACKUP
ST5	Current_key: EXPIRED, New_key: NON

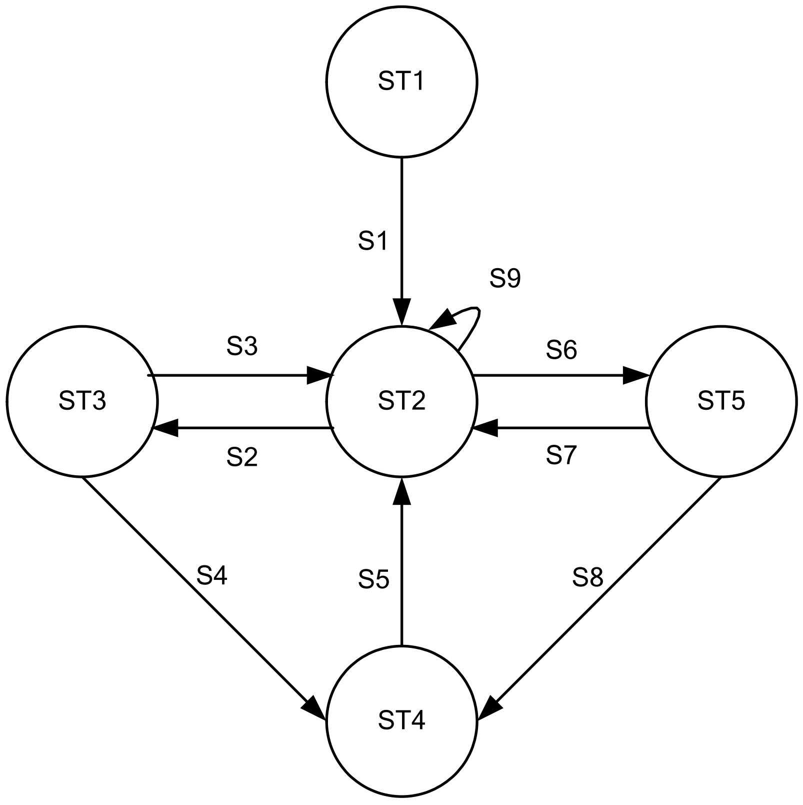

Figure 116 – Key update state machine for FD

Table 115 – Key update state transition

#	Current State	Event or condition => action	Next state

S1	ST1	ASN == New_key.KeyActiveSlot
		=>
		Current_key:= New_key,
		Current_key.state:= USING
			ST2

S2	ST2	Reception of New_key && ASN < New_key.KeyActiveSlot => New_key.state:= BACKUP	ST3

S3	ST3	ASN == New_key.KeyActiveSlot
		=>
		Current_key.state:= INVALID,
		Current_key:= New_key,
		Current_key.state:= USING
			ST2

S4	ST3	ASN > Current_key.KeyActiveSlot + KeyUpdateDur && ASN < New_key.KeyActiveSLot => Current_key.state:= EXPIRED,	ST4

S5	ST4	ASN == New_key.KeyActiveSlot
		=>
		Current_key.state:= INVALID,
		Current_key:= New_key,
		Current_key.state:= USING
			ST2

S6	ST2	ASN > Current_key.KeyActiveSlot + KeyUpdateDur && No New_key => Current_key.state:= EXPIRED	ST5

S7	ST5	Reception of New_key && ASN >= New_key.KeyActiveSlot
		=>
		Current_key.state:= INVALID,
		Current_key:= New_key,
		Current_key.state:= USING
			ST2

S8	ST5	Reception of New_key && ASN < New_key.KeyActiveSlot => New_key.state:= BACKUP	ST4

S9	ST2	Reception of New_key && ASN >= New_key.KeyActiveSlot
		=>
		Current_key.state:= INVALID,
		Current_key:= New_key,
		Current_key.state:= USING
			ST2

11.5 DLL secure communication

DLL shall perform encryption/decryption and integrity check on DLL frames according to the value of SecLevel in MIB.

The encryption/decryption and integrity check are performed by CCM* (see IEEE STD 802.15.4-2011). The secure DLL frame format is shown in Figure 117.

When performing encryption/decryption and integrity check, different keys shall be used according to the phase and the frame type shown in Table 116.

Table 116 – Keys used in DLL secure communication

Phase	Frame type	Key type
Before the establishment of KEDU and KEDB	All frames needs secure protection	KS
After the establishment of KEDU and KEDB	Unicast frame	KEDU
	Non-aggregation broadcast frame	KEDB
	Aggregation broadcast frame	KEDU is used for the frame sent to corresponding field device in the aggregation broadcast frame; KEDB is used for the whole aggregation broadcast frame.

Before access device sends an aggregation broadcast frame, DLL encrypts and/or calculates MIC for each frame inside the aggregation broadcast frame using their related KEDUs separately, and then, DLL encrypts and/or calculates MIC for the whole aggregation broadcast frame using KEDB. After receiving an aggregation broadcast frame, DLL of field device decrypts and/or checks integrity for the whole aggregation broadcast frame using KEDB, and then, decrypts and/or checks integrity its own frame in the aggregation broadcast frame using KEDU.

11.6 Security alarm

WIA-FA defines two types security alarm:

– Key attacked alarm: the count of the key being attacked is over MaxKeyAttackedNum (see 6.7.1.2.1) in a AttackStatisDur (see 6.7.1.2.1);
– Key update timeout alarm: the key is used over KeyUpdateDur (see 6.7.1.2.1) period and there is no new key available for update.

Once the security management module of field device detects a security alarm event, it shall set the corresponding bit of the AlarmFlag (see Table 20) of the related key to 1, then invoke the security alarm request primitive to report the security alarm event and other security alarm events existing in the field device to the security manager in the gateway device periodically according to the AlarmRptDur (see 6.7.1.2.1).

Once the security manager in the gateway device receives a security alarm request, it shall update the keys corresponding to the security alarms in the security alarm request immediately.

Once the key of which the AlarmFlag is not equal to 0 is updated successfully, its AlarmFlag shall be cleared.

11.7 Secure frame format

11.7.1 General secure DLL frame format

Figure 117 shows the general format of secure DLL frame.

DLPDU			
7/8 octets	**Variable length**	**0/4/8/16 octets**	**2 octets**
DLL frame header	DLL payload	MIC	FCS

Figure 117 – General secure DLL frame format

– DLL frame header, see 8.4.1;

– DLL payload: according to the value of SecLevel in MIB, if encryption is enable, DLL payload shall be encrypted;

– MIC: according to the value of SecLevel in MIB, if integrity check is enable, a message integrity code is generated to protect DLL frame header and DLL payload, and filled to MIC;

– FCS.

Table 117 shows the available security levels for DLL in WIA-FA.

Table 117 – Available security levels for DLL

SecLevel	Encryption	MIC length (bits)
0/1	Disable	0
2	Disable	32 (MIC-32)
3	Disable	64 (MIC-64)
4	Disable	128 (MIC-128)
5	Enable	0
6	Enable	32 (MIC-32)
7	Enable	64 (MIC-64)
8	Enable	128 (MIC-128)

Corresponding to the NONCE format defined in Figure 109, PhyAddress field is set to the 64 bits physical address of the field device in unicast communication, and is set to 0 in broadcast communication; TimeStamp field is set to the low 4 bytes of the ASN on which the frame is transmitted or received; SecurityLevel field is set to the value of SecLevel in MIB.

11.7.2 Secure aggregation frame format

Figure 118 shows the format of secure aggregation frame.

		The first frame				**…**	**The n^th frame**					
7/8 octets	**1 octet**	**2 octets**	**1 octet**	**Variable length**	**0/4/8/16 octets**	**…**	**2 octets**	**1 octet**	**Variable length**	**0/4/8/16 octets**	**0/4/8/16 octets**	**2 octets**
DLL frame header	Aggregated number	Field device address	Data length	Data	sMIC	…	Field device address	Data length	Data	sMIC	MIC	FCS

Figure 118 – Secure aggregation frame format

- DLL frame header: see 8.4.1;
- Aggregation number: see 8.1.5;
- Field device address: see 8.1.5;
- Data length: see 8.1.5;
- Data: see 8.1.5 according to the value of SecLevel in MIB, if encryption is enable, Data shall be encrypted with the KEDU of the field device related to the frame;
- sMIC: according to the value of SecLevel in MIB, if integrity check is enable, a message integrity code is generated with the KEDU of the field device related to the frame to protect the Dest. Addr., the Data length and the Data of the frame in the aggregation frame, and filled to sMIC of the frame;
- MIC: see 11.7.1, generated with the KEDB;
- FCS.

For the protection of each frame in the aggregation frame, the PhyAddress field in NONCE is set to the 64 bits physical address of the related field device; and for the protection of the whole aggregation frame, the PhyAddress field in NONCE is set to 0.

11.7.3 Key establish request frame format

Figure 119 shows the format of the key establish request frame.

7/8 octets	29 octets	0/4/8/16 octets	2 octets
DLL frame header	KeyMaterial	MIC	FCS

Figure 119 – Key establish request frame format

- DLL frame header, see 8.4.1;
- KeyMaterial: key material for key establish, defined in Table 104;
- MIC: see 11.7.1;
- FCS.

11.7.4 Key establish response frame format

Figure 120 shows the format of the key establish response frame.

7/8 octets	2 octets	1 octet	0/4/8/16 octets	2 octets
DLL frame header	KeyID	Status	MIC	FCS

Figure 120 – Key establish response frame format

- DLL frame header: see 8.4.1;
- KeyID: the identifier of the key established;
- Status: the result of key establish, see the parameter Status in Table 106;
- MIC: see 11.7.1;
- FCS.

11.7.5 Key update request frame format

Figure 121 shows the format of the key update request frame.

7/8 octets	29 octets	0/4/8/16 octets	2 octets
DLL frame header	KeyMaterial	MIC	FCS

Figure 121 – Key update request frame format

- DLL frame header, see 8.4.1;
- KeyMaterial: key material for key update, defined in Table 104;
- MIC: see 11.7.1;
- FCS.

11.7.6 Key update response frame format

Figure 122 shows the format of the key update response frame.

7/8 octets	**2 octets**	**1 octet**	**0/4/8/16 octets**	**2 octets**
DLL frame header	KeyID	Status	MIC	FCS

Figure 122 – Key update response frame format

- DLL frame header: see 8.4.1;
- KeyID: the identifier of the key for update;
- Status: the result of key for update, see the parameter Status in Table 110;
- MIC: see 11.7.1;
- FCS.

11.7.7 Security alarm request frame format

Figure 123 shows the format of the security alarm request frame.

7/8 octets	**1 octet**	**Variable length**	**0/4/8/16 octets**	**2 octets**
DLL frame header	SecAlarmCount	SecAlarmList	MIC	FCS

Figure 123 – Security alarm request frame format

- DLL frame header: see 8.4.1;
- SecAlarmCount: the count of security alarms contained in SecAlarmList, defined in Table 112;
- SecAlarmList: the list of security alarms, defined in Table 112;
- MIC: see 11.7.1;
- FCS.

Annex A
(informative)

Security strategy for WIA-FA network

A.1 Risk analysis for WIA-FA network

As an open system, there are potential inevitable security risks in WIA-FA network. Therefore, the necessary security measures must be applied to protect the resources within the system and maintain the normal production. The main goal of WIA-FA network security is to protect the normal operation of the system, to detect attacks in time and respond in time, ensuring the safety and minimizing the loss.

The data transmission of WIA-FA network is vulnerable to eavesdrop, manipulation or replay attacks. These threats can be divided into two kinds, malicious and non-malicious, including accessing and manipulating data or information without authentication by using network resource and DoS attack. There are three threats: risk from the outside of company, risk from the management network of the company and risk from internal WIA-FA network itself.

A.2 Security principles for WIA-FA network

According the characteristics of the WIA-FA network, the following security principles are recommended:

- Easy to deployment and use;
- Minimize human-related operations;
- Extend battery life, such as reducing the frame size, using low power encryption technologies, etc.;
- Maximize the use of existing encryption and authentication technologies and existing standards.

A.3 Security objectives for WIA-FA network

The objectives of security in the WIA-FA network include:

- System availability, which refers to ensure the access of the system resources when needed by the legal users;
- Data integrity, which refers to maintaining and assuring the accuracy and consistency of the information;
- Device authentication, which is used to authenticate a device;
- Confidentiality, which refers to ensure that the system hardware, software, and data can be accessed by legal users only;
- Key management, which refers to key establishment and key update.

A.4 Security grade of WIA-FA network

The grades of security and measures are shown in Table A.1.

Table A.1 – Security grades for WIA-FA network

WIA-FA network security level	**Grade 0**	**Grade 1**	**Grade 2**
Security measure	-FCS	-FCS -Device authentication	-FCS -Device authentication -Encryption and Message Integrity Check

Annex B
(informative)

Regional modification for compliance with ETSI standards

B.1 General

WIA-FA restricts the usage of the spectrum to the 2,4 GHz ISM band, see Clause 7.

Additional requirements apply in Europe to wide band radio frequency transmitting equipment operating in this 2,4 GHz ISM band. Some of these European requirements can be met by complying with two Harmonized Standards from ETSI, EN 300 328 and EN 300 440-2.

This Annex B provides additional requirements for compliance of WIA-FA devices operating in the 2,4 GHz ISM band with these two ETSI standards:

- EN 300 440-2 is applicable for equipment (devices) with a maximum transmit power between 0 dBm and 10 dBm e.i.r.p. (see Clause B.2);
- EN 300 328 is applicable for equipment (devices) with a maximum transmit power between 10 dBm and 100 dBm e.i.r.p. (see Clause B.3).

NOTE In this Annex B, the term “devices” refers to electronics with radios operating according to the appropriate standard; these include but are not limited to gateway devices, access devices and field devices.

B.2 Compliance with ETSI EN 300 440-2 V1.4.1

Table B.1 specifies the additional requirements which allow WIA-FA devices operating in the 2,4 GHz ISM band to satisfy the transmit power limitation requirements of EN 300 440-2 V1.4.1.

NOTE EN 300 440-2 V1.4.1 is listed as a Harmonized Standard under the Radio Equipment Directive 2014/53/EU.

Table B.1 – Applicable EN 300 440-2 requirements list

Parameter	EN 300 440-2 V1.4.1 requirements	Additional requirements
Maximum Transmit Power	10 dBm (10 mW) e.i.r.p.	Devices shall be configured to emit less than 10 dBm e.i.r.p., that means less than 10 dBm electrical power with an antenna gain of 1. If the antenna gain is different to 1, then the resulting e.i.r.p. shall be less than 10 dBm by an adequate electrical power adjustment. This requirement overwrites the requirement about electrical power given in 7.3.4.

B.3 Compliance with ETSI EN 300 328 V1.8.1

Table B.2 specify the additional requirements which allow WIA-FA devices operating in the 2,4 GHz ISM band to satisfy various requirements of EN 300 328 V1.8.1.

NOTE EN 300 328 V1.8.1 is listed as a Harmonized Standard under the Radio Equipment Directive 2014/53/EU.

Table B.2 – Applicable EN 300 328 requirements list

Parameter	EN 300 328 V1.8.1 requirements	Additional requirements
Maximum Transmit Power	20 dBm (100 mW) e.i.r.p.	Devices shall be configured to emit less than 10 dBm ± 3 dBm e.i.r.p. according to 7.3.4.
Maximum Power Spectral Density	10 mW/MHz e.i.r.p.	The maximum e.i.r.p. of 13 dBm results in a spectral density below 10 mW/MHz as required.
Duty Cycle, Tx-sequence, Tx-gap	For non-adaptive equipment with maximum e.i.r.p. above 10 dBm: Maximum Tx-Sequence Time = Minimum Tx-gap Time = M 3,5 ms < M < 10 ms	The Maximum Tx-sequence Time for WIA-FA is identical to maximum transmission time; the Minimum Tx-gap Time for WIA-FA is identical to maximum transmission time. The Duty Cycle of WIA-FA shall be less than 50 %.
Medium Utilization Limit	For non-adaptive equipment with maximum e.i.r.p. above 10 dBm: Maximum Medium Utilization factor = 10 %.	The Duty Cycle of a WIA-FA device cannot exceed 50 %. This results in a MU factor value of less than 10 % as required. MU = 13 dBm e.i.r.p. ×50 % <10 %

For WIA-FA, the transmissions of unicast data, broadcast data, unicast ACK, and broadcast NACK are respectively using one separate timeslot. After transmitting a data that required ACK by a source device, the destination device shall not return ACK during the same timeslot. The ACK is returned by using aother timeslot that is carefully scheduled. The transmission method of NACK is the same as that of ACK. In brief, the transmission of an ACK or a NACK is the same as that of general data. The timeslot timing template of WIA-FA is defined in Figure B.1. Timeslot timing definitions and calculations are shown in Table B.2.

The maximum transmission time for a WIA-FA packet is listed in Table B.3, Table B.4, Table B.5, and Table B.6 according to different physical layer. The Maximum Tx-sequence Time (defined in ETSI EN 300 328 V1.8.1) for WIA-FA is identical to the maximum transmission time (TxMaxPHYPacket in Figure B.1); the Minimum Tx-gap Time (defined in ETSI EN 300 328 V1.8.1) for WIA-FA is identical to the maximum transmission time (TxMaxPHYPacket in Figure B.1) compliant with ETSI EN 300 328 V 1.8.1 requirements.

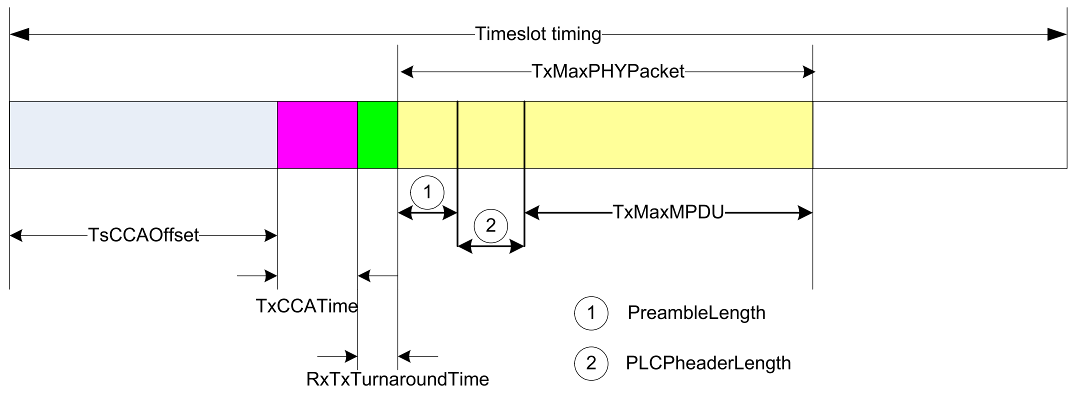

Figure B.1 – Timeslot timing template

Table B.3 – Timeslot timing definitions and calculations

Symbol	Definition	FHSS Required value	DSSS/HR-DSSS Required value	OFDM Required value		
				20M	10M	5M
TsCCAOffset	Start of slot to beginning of CCA	–	–	–	–	–
TsCCATime	Time to perform CCA (8 symbols)	≤27 μs	≤15 μs	<4 μs	<8 μs	<16 μs
RxTxTurnaroundTime	The longer of the time it takes to switch from reception to transmission or vice versa	20 μs	≤5 μs	≤2 μs	≤2 μs	≤2 μs
PreambleLength	Preamble transmission time	96 μs	144 μs	16 μs	32 μs	64 μs
PLCPheaderLength	PLCP Header transmission time	32 μs	48 μs	4 μs	8 μs	16 μs
TxMaxMPDU [a]	The longest time to transmit a MPDU.	Table A.2	Table A.3	Table A.4		
TxMaxPHYPacket [b]	PreambleLength+ PLCPheaderLength+ TxMaxMPDU					

[a] TxMaxMPDU =(Maximum length of a MPDU×8) bit /Data rate×1 000×1 000 b/s

[b] TxMaxPHYPacket = PreambleLength+ PLCPheaderLength+ TxMaxMPDU

Table B.4 – TxMaxPHYPacket of FHSS

Rate (Mb/s)	TxMaxPHYPacket (μs) (MPDU:1 000 Octets)
1	8 000+128=8 128 [a,b]
1,5	5 333+128=5 461
2	4 000+128=4 128
2,5	3 200+128=3 328
3	2 667+128=2 795
3,5	2 286+128=2 414
4	2 000+128=2 128
4,5	1 778+128=1 906

[a] PreambleLength+ PLCPheaderLength = 96 +32 =128 μs

[b] TxMaxPHYPacket = PreambleLength+ PLCPheaderLength+ TxMaxMPDU

Table B.5 – TxMaxPHYPacket of DSSS/HR-DSSS

Rate (Mb/s)	**TxMaxPHYPacket (μs)** **(MPDU:1 000 Octets)**
1	8 000+192=8 192 [a,b]
2	4 000+192=4 192
5,5	1 455+192=1 647
11	727+192=919

[a] PreambleLength+ PLCPheaderLength = 144 +48 =192 μs

[b] TxMaxPHYPacket = PreambleLength+ PLCPheaderLength+ TxMaxMPDU

Table B.6 – TxMaxMPDU of OFDM

Rate (Mb/s)	**TxMaxPHYPacket (μs)** **(MPDU:1 000 Octets)**		
	20M	**10M**	**5M**
6	1 333+20=1 353 [a1,b]	1 333+40=1 373[a2,b]	1 333+80=1 413[a3,b]
9	889+20=909	889+40=929	889+80=969
12	667+20=687	667+40=707	667+80=747
18	444+20=464	444+40=484	444+80=524
24	333+20=353	333+40=373	333+80=413
36	222+20=242	222+40=262	222+80=302
48	167+20=187	167+40=207	167+80=247
54	148+20=168	148+40=188	148+80=188

[a1] For 20M of OFDM, PreambleLength+ PLCPheaderLength =16+4=20 μs

[a2] For 20M of OFDM, PreambleLength+ PLCPheaderLength =32+8=40 μs

[a3] For 20M of OFDM, PreambleLength+ PLCPheaderLength =64+16=80 μs

[b] TxMaxPHYPacket = PreambleLength+ PLCPheaderLength+ TxMaxMPDU

Bibliography

[1] ISO/IEC/IEEE 60559:2011, *Information technology – Microprocessor Systems – Floating-Point arithmetic*

[2] IEC 62657-2:2013, *Industrial communication networks – Wireless communication Networks – Part 2: Coexistence management*

[3] 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)*

[4] EN 300 328, *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 requirement sof article 3.2 of the R&TTE Directive*

[5] EN 300 440-2, *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*

[6] Handbook of Applied Crytography, A. Menezes, P. van Oorschot, and S. Vanstone, CRC Press, 1996