IEC 62714-1

Edition 1.0 2014-06

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Engineering data exchange format for use in industrial automation systems engineering – Automation markup language – Part 1: Architecture and general requirements

Format d'échange de données techniques pour une utilisation dans l'ingénierie des systèmes d'automatisation industrielle – Automation markup language – Partie 1: Architecture et exigences générales

IEC 62714-1:2014-06(en-fr)

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

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

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

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

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

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 It also gives information on projects, replaced and withdrawn publications.

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

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

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

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

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

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

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

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

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

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

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

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

IEC 62714-1

Edition 1.0 2014-06

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Engineering data exchange format for use in industrial automation systems engineering – Automation markup language – Part 1: Architecture and general requirements

Format d'échange de données techniques pour une utilisation dans l'ingénierie des systèmes d'automatisation industrielle – Automation markup language – Partie 1: Architecture et exigences générales

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

PRICE CODE
CODE PRIX XC

ICS 25.040.40; 35.06; 35.240.50

ISBN 978-2-8322-1554-8

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

ENGINEERING DATA EXCHANGE FORMAT FOR USE IN INDUSTRIAL AUTOMATION SYSTEMS ENGINEERING – AUTOMATION MARKUP LANGUAGE –

Part 1: Architecture and general requirements

FOREWORD

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

International Standard IEC 62714-1 has been prepared by subcommittee 65E: Devices and integration in enterprise systems, of IEC technical committee 65: Industrial-process measurement, control and automation.

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

FDIS	Report on voting
65E/385/FDIS	65E/396/RVD

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

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

A list of all parts in the IEC 62714 series, published under the general title *Engineering data exchange format for use in industrial automation systems engineering – Automation Markup Language*, can be found on the IEC website.

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

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

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

INTRODUCTION

IEC 62714 is a solution for data exchange focusing on the domain of automation engineering.

The data exchange format defined in the IEC 62714 series (Automation Markup Language, AML) is an XML schema based data format and has been developed in order to support the data exchange in a heterogeneous engineering tools landscape.

The goal of AML is to interconnect engineering tools in their different disciplines, e.g. mechanical plant engineering, electrical design, process engineering, process control engineering, HMI development, PLC programming, robot programming, etc.

AML stores engineering information following the object oriented paradigm and allows modelling of physical and logical plant components as data objects encapsulating different aspects. An object may consist of other sub-objects, and may itself be part of a larger composition or aggregation. Typical objects in plant automation comprise information on topology, geometry, kinematics and logic, whereas logic comprises sequencing, behaviour and control. Therefore, an important focus in the data exchange in engineering is the exchange of object oriented data structures, geometry, kinematics and logic.

AML combines existing industry data formats that are designed for the storage and exchange of different aspects of engineering information. These data formats are used on an “as-is” basis within their own specifications and are not branched for AML needs.

The core of AML is the top-level data format CAEX that connects the different data formats. Therefore, AML has an inherent distributed document architecture.

Figure 1 illustrates the basic AML architecture and the distribution of topology, geometry, kinematics and logic information.

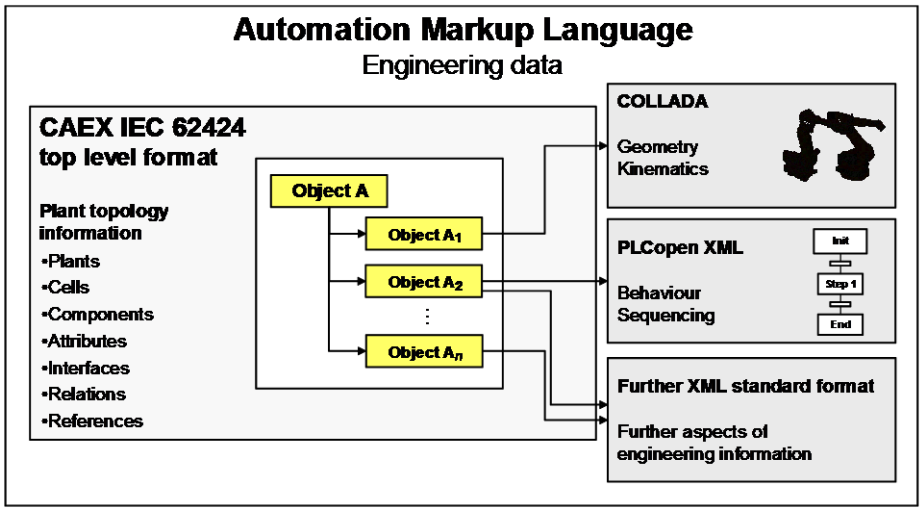


Figure 1 – Overview of the engineering data exchange format AML

Due to the different aspects of AML, the IEC 62714 series consists of different parts focussing on different aspects:

- IEC 62714-1: Architecture and general requirements

  This part specifies the general AML architecture, the modelling of engineering data, classes, instances, relations, references, hierarchies, basic AML libraries and extended AML concepts. It is the basis of all future parts, and it provides mechanisms to reference other sub formats.

- IEC 62714-2: Role class libraries

  This part is intended to specify additional AML libraries.

- IEC 62714-3: Geometry and kinematics

  This part is intended to specify the modelling of geometry and kinematics information.

- IEC 62714-4: Logic

  This part is intended to specify the modelling of logics, sequencing, behaviour and control related information.

Further parts may be added in the future in order to interconnect further data standards to AML.

As long as no further parts describe the integration of further standards, it is important to focus on a limited set of sub data formats. Otherwise it would open up the usage of any data format and data exchange would not work.

Annex A gives an informative introduction, use cases and examples regarding AML.

Annex B gives an informative XML representation of the libraries defined in this part of IEC 62714.

ENGINEERING DATA EXCHANGE FORMAT FOR USE IN INDUSTRIAL AUTOMATION SYSTEMS ENGINEERING – AUTOMATION MARKUP LANGUAGE –

Part 1: Architecture and general requirements

1 Scope

This part of IEC 62714 specifies general requirements and the architecture of AML for the modelling of engineering information which is exchanged between engineering tools for industrial automation and control systems. Its provisions apply to the export/import applications of related tools.

This part of IEC 62714 does not define details of the data exchange procedure or implementation requirements for the import/export tools.

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 62424:2008, *Representation of process control engineering – Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools*

IEC 62714 (all parts), *Engineering data exchange format for use in industrial automation systems engineering – Automation Markup Language*

ISO/IEC 9834-8, *Information technology – Open Systems Interconnection – Procedures for the operation of OSI Registration Authorities: Generation and registration of Universally Unique Identifiers (UUIDs) and their use as ASN.1 Object Identifier components*

ISO/PAS 17506, *Industrial automation systems and integration — COLLADA digital asset schema specification for 3D visualization of industrial data*

COLLADA 1.4.1:March 2008, COLLADA – Digital Asset Schema Release 1.4.1
(available at <http://www.khronos.org/files/collada_spec_1_4.pdf>)

Extensible Markup Language (XML) 1.0 1.0:2004, W3C Recommendation
(available at <http://www.w3.org/TR/2004/REC-xml-20040204/>)

PLCopen XML 2.0:December 3rd 2008 and PLCopen XML 2.0.1:May 8th 2009, XML formats for *IEC 61131*-3
(available at <*http://www.plcopen.org*>/)

3 Terms, definitions and abbreviations

3.1 Terms and definitions

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

3.1.1
AML
XML based data exchange format for plant engineering data following IEC 62714

3.1.2
automation object
physical or logical entity in the automated system

Note 1 to entry: An example of an automation object is an automation component, a valve or a signal.

3.1.3
AML object
data representation of an automation object with a relation to an AML role class

Note 1 to entry: The AML objects are the core elements of AML. They represent instances and may contain administration items, attributes, interfaces, relations and references.

3.1.4
AML class
predefined AML object type

Note 1 to entry: AML classes are stored within AML libraries.

Note 2 to entry: AML classes define reusable sample solutions, characterized by attributes, interfaces and aggregated objects.

Note 3 to entry: AML classes can be used for multiple instantiations.

3.1.5
AML attribute
property which belongs to an AML object

Note 1 to entry: AML attributes are described as an XML element corresponding to IEC 62424:2008, A.2.4.

3.1.6
AML document
certain CAEX document following IEC 62714 including all referenced sub documents

Note 1 to entry: AML documents may be stored as files, but also e.g. as string or data streams.

3.1.7
AML file
certain CAEX file following IEC 62714-1 with the extension .aml excluding all referenced sub files

3.1.8
AML interface
single connection point that belongs to an AML object and can be linked with another interface

Note 1 to entry: Interfaces allow the description of relations between objects by the definition of CAEX InternalLinks. Examples are a signal interface, a device interface or a power interface.

3.1.9
AML library
library containing AML classes

3.1.10
AML Port
AML object that represents a container for a group of interfaces characterized by additional properties

Note 1 to entry: Ports belong to a parent AML object and describe complex interfaces of this object. Ports can be connected to each other on a higher abstraction level.

3.1.11
AML Group
AML object providing a certain view on AML objects

3.1.12
AML Facet
AML object providing a certain view on AML attributes or interfaces of one AML object

3.1.13
CAEX
neutral XML based data format

Note 1 to entry: CAEX is a neutral data format according to IEC 62424:2008, Clause 7, Annex A and Annex C

3.1.14
copy-instance-relation
relation between the instance and the corresponding class where the instance is created by copying the class data structures

Note 1 to entry: The instance receives a copy of all features and properties of the source AML class. Modifications of the class do not lead to modifications of the instance. Within the instance, class properties are individualized. Further copies are possible due to the knowledge of the source AML class.

3.1.15
universal unique identifier
UUID
unique identifier for AML objects

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

3.1.16
global unique identifier
GUID
implementation of a UUID

Note 1 to entry: Real GUID example: “{AC76BA86-7AD7-1033-7B44-A70000000000}”.

Note 2 to entry: In IEC 62714, GUIDs are also presented in a short form such as “GUID1”, “GUID2” etc. This serves the readability and acts as a real GUID.

Note 3 to entry: This note applies to the French language only.

3.1.17
inheritance relation
relation between two AML classes

Note 1 to entry: The derived class inherits all attributes and features of the parent class.

3.1.18
instance
data representation of an individual physical or logical item

Note 1 to entry: Instances can be extended, e.g. by aggregated objects or attributes.

3.1.19
PropertySet
AML standard role class containing a set of semantically predefined attributes

3.1.20
topology
hierarchical structure of a system, visualizable as object tree

Note 1 to entry: Multiple hierarchies, crossed structures and object networks are included.

3.1.21
plant topology
hierarchical structure of a plant, visualizable as object tree

3.1.22
publish, verb
to model a data structure of an external document for usage within CAEX

Note 1 to entry: This allows definition of relations between data structures of independent external documents.

3.1.23
relation
association between CAEX objects

Note 1 to entry: Examples for relations are parent-child-relations and class-instance-relations.

3.1.24
link
connection between objects of type CAEX ExternalInterface

Note 1 to entry: A link is modelled by means of CAEX InternalLink.

3.1.25
reference
association between a CAEX InternalElement and externally stored information

3.2 Abbreviations

Table 1 – Abbreviations

AML	Automation Markup Language
CAE	Computer Aided Engineering
CAEX	Computer Aided Engineering eXchange
COLLADA	Collaborative design activity
GUID	Global unique identifier
HMI	Human machine interface
ID	Identifier
MES	Manufacturing execution system
PLC	Programmable logic controller
URL	Uniform resource locator
URI	Uniform resource identifier
UUID	Universal unique identifier
XML	Extensible Markup Language

4 Conformity

To claim conformity to this part of IEC 62714 with respect to the support of AML, the requirements of Clauses 5, 6, 7 and 8 shall be fulfilled.

5 AML architecture specification

5.1 General

The centre of AML is the top-level data format CAEX, a neutral data format according to IEC 62424:2008, Clause 7, Annex A and Annex C, that interconnects established data formats for the engineering aspects for topology, geometry, kinematics, behaviour and sequencing information. Therefore, a basic characteristic of AML is an inherent distributed document architecture focussing on the above mentioned engineering aspects.

Figures are illustrative only. The graphical representation is not normative.

5.2 General AML architecture

Regarding the general AML architecture, the following provisions apply:

Plant topology information: The plant topology acts as the top-level data structure of the plant engineering information and shall be modelled by means of the data format CAEX according to IEC 62424:2008, Clause 7, Annex A and Annex C. Semantic extensions of CAEX are described separately. Multiple and crossed hierarchy structures shall be used by means of the mirror object concept according to IEC 62424:2008, A.2.14. Mirror objects shall not be modified; all changes shall be done at the master object.

NOTE 1 According to IEC 62424:2008, A.2.14, an AML object with a relation to another AML object is called “mirror object” whereas the related AML object is called “master object”. The mirror object is considered to be identical to the master object. This enables placing one object instance into different plant hierarchies and thus allows modelling of complex object networks with crossed structures.

NOTE 2 IEC 62714 does not syntactically modify the CAEX data format. An informative overview and additional examples regarding the plant topology are provided in A.1.2 and in IEC 62424:2008, Annex D.

Reference and relation information: References and relations shall be stored according to 5.6 and 5.7. Relations between externally stored information shall be stored with CAEX mechanisms. If required, the related link partners shall be published in the CAEX plant topology description by means of CAEX ExternalInterfaces. They shall be derived from AML standard interface classes specified in 6.3.

NOTE 3 References depict links from CAEX objects to externally stored information. An informative overview about relations is provided in A.1.5. References and publishing of interfaces are described in additional parts of IEC 62714.

NOTE 4 Relations depict associations between CAEX objects.

Geometry and kinematics information: Geometry and kinematics relevant information shall be stored using the data format COLLADA™[1]. COLLADA interfaces that need to be interconnected within the top level format shall be published as CAEX ExternalInterfaces.

NOTE 5 IEC 62714 does not syntactically modify the COLLADA data format. An overview example of how to reference COLLADA is provided in A.1.3. Details are intended to be specified in IEC 62714-3.

NOTE 6 By means of the COLLADA geometry information of different objects, a complete scene can be derived automatically. These files can be referenced from CAEX and can be interlinked using CAEX linking mechanisms.

Logic information: Logic information shall be stored using the data format PLCopen XML. If logic items, e.g. variables or signals, need to be interconnected within the top level format, they shall be published as CAEX ExternalInterfaces. All items of PLCopen XML that are published within the top level format shall have a unique ID within PLCopen XML.

---

1 COLLADA is the trademark of a product supplied by the Khronos Group. This information is given for the convenience of users of this standard and does not constitute an endorsement by IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

NOTE 7 Logic information describes sequences of actions and the internal behaviour of objects including I/O connections and logical variables. IEC 62714 does not modify the PLCopen XML format. An informative overview of how to reference logic information is provided in A.1.4. Details are intended to be specified in IEC 62714-4.

Referencing other data formats: IEC 62714 may be extended in the future by additional parts specifying the integration of further XML data formats utilizing the AML reference mechanisms. Details may be defined in additional parts of IEC 62714.

The data format AML does not provide consistency checks of constraints, attribute values, relations, references or the semantic correctness of the contained data: this is the responsibility of the source or target tool or the corresponding import/export application. AML only allows a syntactical proof of the document against the corresponding schemas.

5.3 AML document versions

Each AML XML document shall store the underlying AML version which this standard follows.

NOTE 1 Normative provisions regarding the version information related to AML object instances are defined in 8.9. The storage of tool specific meta information is defined in 5.4.

Hence, the following provisions apply:

- The CAEX root element “CAEXFile” of each AML top level document shall have the CAEX child element “AdditionalInformation”.
- The element “AdditionalInformation” shall have an attribute “AutomationMLVersion”.
- The value of this attribute “AutomationMLVersion” shall be stored in the XML document. It shall be “2.0” to correspond to this standard.
- Every referenced CAEX document shall follow the same AML version of the root document. Mixing of documents with different AML versions is explicitly forbidden.
- Every referenced external document shall also follow the named schema versions specified in the above AML version specification. Mixing of external document versions outside of one AML version specification is explicitly forbidden.

Figure 2 illustrates the XML text for a CAEX document following the AML version 2.0.

```
<CAEXFile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="
CAEX_ClassModel.xsd" FileName="AutomationMLStandardLibrary2010-01-14_v1.99.aml" SchemaVersion="2.15">
  <AdditionalInformation AutomationMLVersion="2.0"/>
```

Figure 2 – AML document version information

- Every AML standard library and every user defined AML library shall define its version number utilizing the CAEX element “Version”. The syntax of the value of the version number is not defined in this part of IEC 62714.
- If required, CAEX classes shall define their version number utilizing the CAEX element “Version”. The syntax and semantic of the version number of classes within an AML library is not defined in this part of IEC 62714.
- Same libraries of different versions are forbidden to be stored in the same AML file.

  NOTE 2 This ensures the uniqueness of AML library names within an AML file.
- The creator of an AML document shall ensure that only version compatible classes and external documents are referenced.

IEC 62714 is based on the following document formats:

- CAEX version 2.15;
- PLCopenXML 2.0 and 2.0.1;
- COLLADA 1.5.0 as specified in ISO/PAS 17506 and COLLADA 1.4.1;

- AML standard libraries as specified in this part of IEC 62714 and further parts of IEC 62714.

5.4 Meta information about the AML source tool

In case of the need of a transfer of user defined data from a source tool to a destination tool, it is necessary to store information about the source tool directly into the AML document. Hence, the following provisions apply:

- Each AML document shall provide information about the tool which has written the AML document.
- In a data exchange tool chain, all participating tools shall store this information in the CAEX document in the same way. Hence, the document may contain information about multiple tools of a data exchange tool chain. A tool may remove the writer information of other tools. This may hinder the iterative data exchange with the other tools: hence the removal of writer information of other tools is not recommended.
- This information shall be stored as part of the CAEX AdditionalInformation of the root object of the CAEX document.
- The AdditionalInfomation block shall be named “WriterHeader”.
- The meta information shall provide information about:
  - the name of the exporting software, the writer tool;
  - the ID of the writer tool (it shall remain unchanged);
  - the vendor of the writer tool;
  - the URL of the writer tool;
  - the product version of the writer tool;
  - the product release number of the writer tool;
  - the last writing time of the writer;
  - the project title of the source project;
  - the project ID of the source project.
- The content of the meta information shall be defined by the writer tool and shall be of type xs:string.
- The required information shall be stored by means of the attributes shown in Table 2.

Table 2 – Meta information about the AML source tool

XML tag name	Type	Level	Example
WriterName	xs:string	Mandatory	“ToolX AML Exporter”
WriterID	xs:string	Mandatory	“ToolXToAML123”
WriterVendor	xs:string	Mandatory	“ToolX Vendor”
WriterVendorURL	xs:string	Mandatory	“http://www.ToolX-Vendor.org”
WriterVersion	xs:string	Mandatory	“0.2”
WriterRelease	xs:string	Mandatory	“123 prealpha”
LastWritingDateTime	xs:DateTime	Mandatory	“2011-05-25T09:30:47”
WriterProjectTitle	xs:string	Optional	“eCarproduction”
WriterProjectID	xs:string	Optional	“eCarproduction_LinePLC.prj”

For the XML representation of the meta information, the following provisions apply:

- The element “WriterHeader” shall be a child XML element of the CAEX element AdditionalInformation of the CAEX root element.

- Each meta information named in Table 2 shall be described as a child XML element of the “WriterHeader”.
- Multiple meta information of the same name are forbidden in the same “WriterHeader” element.
- The order of the meta information shall be equivalent to Table 2.

Figure 3 illustrates the required XML text by means of an example.

```
<AdditionalInformation>
  <WriterHeader>
    <WriterName>ToolX AML Exporter</WriterName>
    <ToolWriterID>ToolXToAML123</ToolWriterID>
    <WriterVendor>ToolX Vendor</WriterVendor>
    <WriterVendorURL>http://www.ToolX-Vendor.org</WriterVendorURL>
    <WriterVersion>0.1</WriterVersion>
    <WriterRelease>123 prealpha</WriterRelease>
    <LastWritingDateTime>2013-12-31T12:00:00</LastWritingDateTime>
    <WriterProjectTitle>eCarproduction</WriterProjectTitle>
    <WriterProjectID>eCarproduction_LinePLC.prj</WriterProjectID>
  </WriterHeader>
</AdditionalInformation>
```

Figure 3 – XML text of the AML source tool information

5.5 Object identification

AML follows the object oriented paradigm. All engineering information is modelled as object or belongs to an object. But, in a heterogeneous tool landscape, different engineering tools use different concepts for the identification of objects, e.g. a unique name, a unique identifier or a unique path. Some tools allow changes of the identifiers over the life time, others do not. IEC 62714 enables the data exchange between different engineering tools with such individual object identification concepts. Owing to the described characteristics, this part of IEC 62714 neutralizes this variety and defines one mandatory object identification concept.

Regarding the object identification, the following provisions apply:

- According to IEC 62424:2008, A.2.2.1, AML classes (RoleClasses, InterfaceClasses and SystemUnitClasses) shall be identified by their CAEX tag “Name”.
- This name shall be unique within the hierarchy level of the corresponding AML library over the life time of the class.
- According to IEC 62424:2008, A.2.8, referencing of classes shall be done via full paths using the corresponding path separators.
- All AML object instances (CAEX InternalElements and CAEX ExternalInterfaces) shall be identified by their CAEX tag “ID”. This identifier shall be a universal unique identifier (UUID) according to ISO/IEC 9834-8.

  NOTE 1 A possible implementation of the UUID is the global unique identifier (GUID).

  NOTE 2 According to IEC 62424:2008, A.3.15, the tag “ID” is not mandatory in contrast to this part of IEC 62714.

  NOTE 3 In this part of IEC 62714, UUIDs are presented in a short form such as “GUID1”, “GUID2” etc.

  NOTE 4 The CAEX tag “Name” is a display name; it has informative character only and can change over the time or tool.
- Once created, this UUID shall never change over the life time of the corresponding object within all participating tools.
- Referencing instances shall use the “ID” value.
- Referencing CAEX interfaces shall be done using the corresponding UUID of the interface’s parent object followed by the separator string “:”and the name of the interface instance.

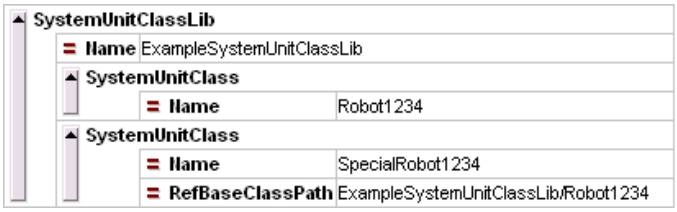
EXAMPLE 1 "GUID:out".

- Referencing CAEX attributes shall be done using the corresponding UUID of the attribute's parent object followed by the separator string "."and the name of the attribute. If the attribute is a nested attribute, the separator string is followed by the sub-path of the attribute.

  EXAMPLE 2 "GUID.Colour".

  EXAMPLE 3 "GUID.Colour.red".

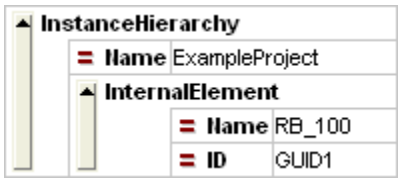
Figure 4 gives an example of a SystemUnitClassLib with the SystemUnitClass "Robot1234" and another SystemUnitClass "SpecialRobot1234" derived from "Robot1234".




```
<SystemUnitClassLib Name="ExampleSystemUnitClassLib">
  <SystemUnitClass Name="Robot1234"/>
  <SystemUnitClass Name="SpecialRobot1234" RefBaseClassPath="ExampleSystemUnitClassLib/Robot1234"/>
</SystemUnitClassLib>
```

Figure 4 – Object identification example of an AML class

Figure 5 gives an example of an InstanceHierarchy with one object "RB_100" which has a unique ID represented by the string "GUID1".




```
<InstanceHierarchy Name="ExampleProject">
  <InternalElement Name="RB_100" ID="GUID1"/>
</InstanceHierarchy>
```

Figure 5 – Object identification example of an AML object instance

5.6 AML relations specification

5.6.1 General

The focus on objects makes it necessary to define a mechanism to set objects in association to each other. This part of IEC 62714 distinguishes between two mechanisms to store this information: references and relations. Subclause 5.6 focuses on relations, whereas 5.7 focuses on references. An informative overview about relations and references is provided in A.1.5.

5.6.2 Parent-child-relations between AML objects

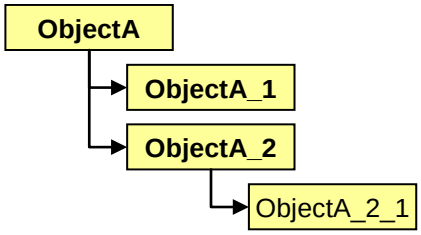
Parent-child-relations between object instances are used to represent hierarchical object structures and describe a "consist-of-relation".

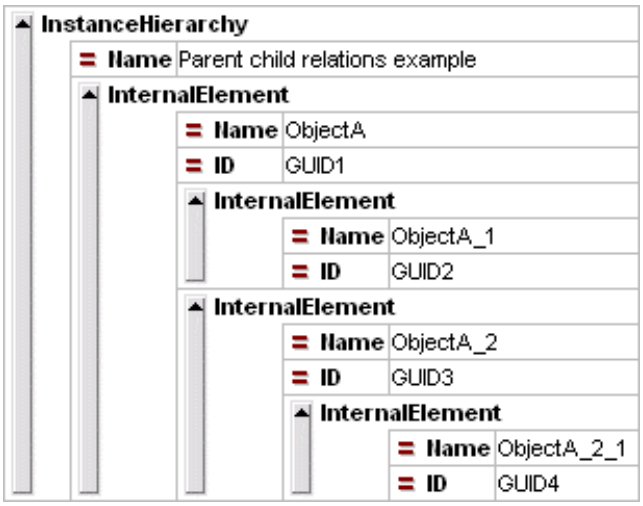
Regarding parent-child-relations between AML objects, the following provision applies:

- The storage of hierarchies shall be done according to IEC 62424:2008, Annex A, e.g. A.2.11.

  NOTE In addition to simple hierarchies, also crossed hierarchies (object networks) can be stored according to IEC 62424:2008, A.2.14.

Figure 6 gives an example of an object hierarchy and its storage.






```
<InstanceHierarchy Name="Parent child relations example">
    <InternalElement Name="ObjectA" ID="GUID1">
        <InternalElement Name="ObjectA_1" ID="GUID2"/>
        <InternalElement Name="ObjectA_2" ID="GUID3">
            <InternalElement Name="ObjectA_2_1" ID="GUID4"/>
        </InternalElement>
    </InternalElement>
</InstanceHierarchy>
```

Figure 6 – Example of a parent-child-relation between AML objects

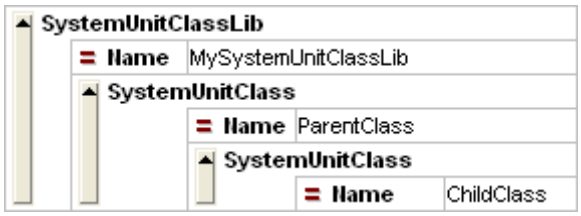
5.6.3 Parent-child-relations between AML classes

Regarding parent-child-relations between AML classes, the following provisions apply:

- A parent-child-relation between AML classes shall describe their hierarchical neighbour ship only. This allows definition of any user-defined hierarchical structure.
- This relation has no further semantics.

NOTE A parent-child-relation does not imply an inheritance relation. Inheritance relations are modelled explicitly as described in 5.6.4.

Figure 7 gives an example of a parent-child-relation between the classes “ParentClass” and “ChildClass”. The “ChildClass” does not have an inheritance relation to its parent.




```
<SystemUnitClassLib Name="MySystemUnitClassLib">
    <SystemUnitClass Name="ParentClass">
        <SystemUnitClass Name="ChildClass"/>
    </SystemUnitClass>
</SystemUnitClassLib>
```

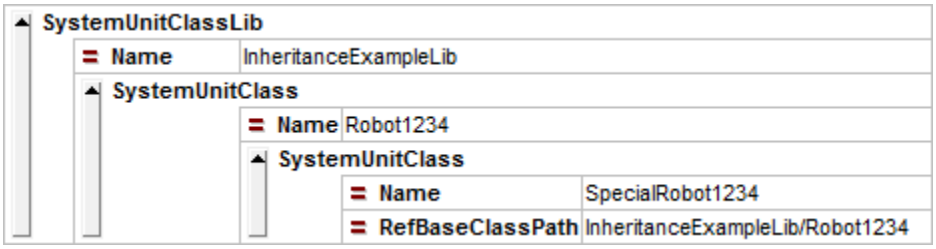
Figure 7 – Example of a parent-child-relation between classes

5.6.4 Inheritance relations

Regarding inheritance relations, the following provisions apply:

- Inheritance between classes shall be defined according to IEC 62424:2008, A.2.7.
- If inheritance is required, the parent class shall be specified using the CAEX tag “RefBaseClassPath” comprising the full path of the class according to IEC 62424:2008, A.2.7.
- If the desired parent class is placed one hierarchy level above the child class, the parent class can be specified by storing the name of the parent class in the CAEX tag “RefBaseClassPath” without providing the full path.

Figure 8 gives an example of the class “Robot1234” and another class “SpecialRobot1234” which inherits from “Robot1234”.




```
<SystemUnitClassLib Name="InheritanceExampleLib">
  <SystemUnitClass Name="Robot1234">
    <SystemUnitClass Name="SpecialRobot1234" RefBaseClassPath="
InheritanceExampleLib/Robot1234"/>
  </SystemUnitClass>
</SystemUnitClassLib>
```

Figure 8 – Example of an inheritance relation between two classes

In addition to this example, the CAEX tag “RefBaseClassPath” can either be “InheritanceExampleLib/Robot1234” as well as “Robot1234” since the parent class is one hierarchy level above the class “SpecialRobot1234”.

5.6.5 Class-instance-relations

Instances are characterized by a unique identifier and parameter set. The following provisions apply:

- An AML object shall be modelled as CAEX InternalElement as part of the CAEX InstanceHierarchy or of a SystemUnitClass.
- An AML object may be a singleton without a relation to any SystemUnitClass.

  NOTE 1 However, an AML object has a relation to a standard AML role class.

  NOTE 2 Instances without a relation to the AutomationMLBaseRole are possible but are user defined objects. They are not AML objects.

- If an AML object has a class-instance-relation to a SystemUnitClass, it shall be created as a copy of this SystemUnitClass including the internal architecture of the class and all inherited information.

  NOTE 3 A SystemUnitClass serves as a template in this way. Changes in the SystemUnitClass are not automatically reflected in the corresponding Automation objects. Furthermore, the Automation object can be transported without the class information; it contains within itself the whole belonging information.

- The extension or reduction of instance data compared to the source class is allowed.

  NOTE 4 The source class is intended to be a suitable starting point for the instance model.

- The copied source class shall be indicated in the CAEX tag “RefBaseSystemUnitPath” of the instance for further usage. This tag shall comprise the full path and name of the source class.

  NOTE 5 If the source-class of an instance changes, this does not entail a change of the instance. The update of instances is a possible tool functionality out of the scope of this part of IEC 62714.

- Changes of a source class should lead to a new version of the class with another name. Within the new class, the full path of the old version of the class should be stored in the CAEX tag “OldVersion”.

  NOTE 6 This provision supports tracking of changes across different versions of a class.

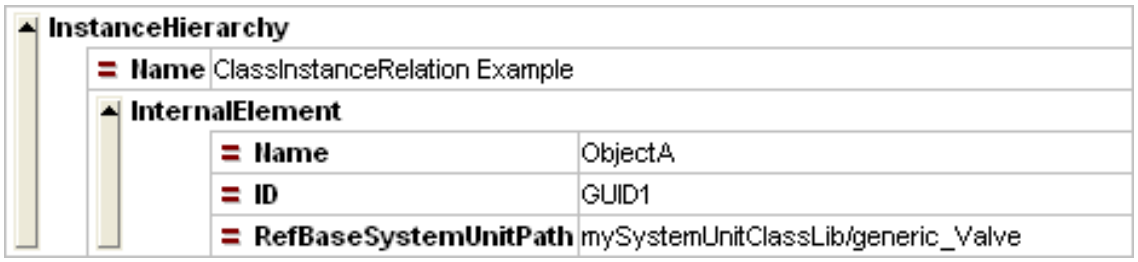
- Inheritance between a SystemUnitClass and an object instance is not allowed.

  NOTE 7 An instance can only be a copy of its class. This is a restriction against IEC 62424:2008, A.2.7. Inheritance between classes and instances can be part of future extension.

- The relation between an instance and a RoleClass shall be indicated according to IEC 62424:2008, A.2.7, by the attribute “RefBaseRoleClassPath” of the belonging RoleRequirement. In contrast to IEC 62424:2008, A.2.7, no inheritance is permitted; all RoleClass specifications shall be copied to the corresponding AML object.

- The relation between a CAEX ExternalInterface and an InterfaceClass shall be indicated according to IEC 62424:2008, A.2.7. In contrast to IEC 62424, no inheritance is allowed; all InterfaceClass specifications shall be copied to the corresponding AML object.

Figure 9 gives an example of a class-instance-relation between the object “ObjectA” and a user-defined SystemUnitClass “generic_Valve”.




```
<InstanceHierarchy Name="ClassInstanceRelation Example">
  <InternalElement Name="ObjectA" ID="GUID1" RefBaseSystemUnitPath="mySystemUnitClassLib/generic_Valve"/>
</InstanceHierarchy>
```

Figure 9 – Example of a class-instance-relation

In addition to the standard class-instance-relation provisions, the following specific provision applies according to the CAEX mirror concept:

- The tag “RefBaseSystemUnitPath” may indicate another object instance instead of a SystemUnitClass according to the mirror concept of IEC 62424:2008, A.2.14.

5.6.6 Instance-instance-relations

Instance-instance-relations are relations between two interfaces of arbitrary AML objects.

Regarding instance-instance-relations, the following provisions apply:

- Instance-instance-relations shall be stored according to IEC 62424:2008, A.2.5.3 and A.2.14, by means of the CAEX InternalLink functionality.
- CAEX InternalLinks should be stored at the CAEX InternalElement which is the lowest common parent of the corresponding connected CAEX objects.
- Instance-instance-relations shall be defined only between CAEX ExternalInterfaces that belong to the corresponding AML objects. This is according to IEC 62424:2008, A.2.3.1.
- The ExternalInterfaces should be derived directly or indirectly from one of the AML standard interface classes.

  NOTE 1 The AML standard interface class library is specified in 6.3. The interface classes define the semantic of the interface and thus the semantic of the link. A link between interfaces without a reference to an interface class has no semantics.

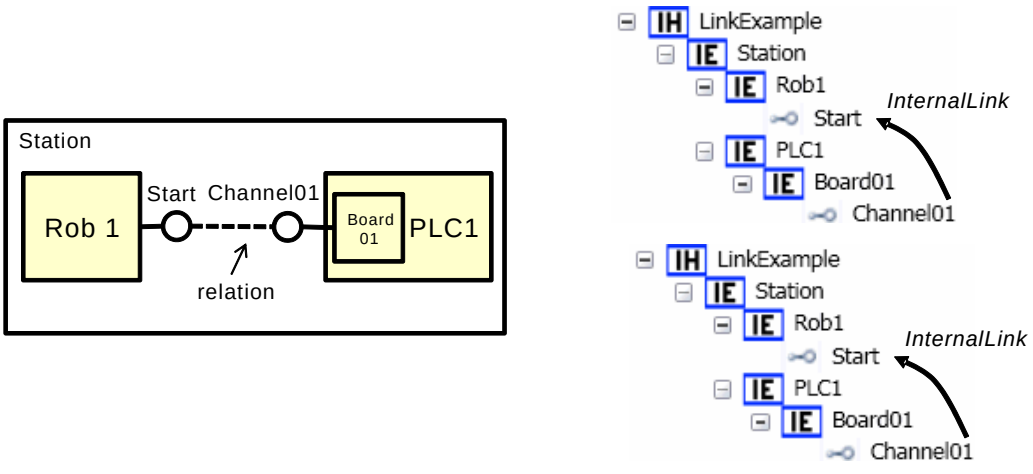
- COLLADA documents may be interlinked. The corresponding COLLADA interfaces may be any items that have a valid URI. If those nodes are required to be interlinked in CAEX, they shall be published in CAEX by adding a CAEX ExternalInterface to the corresponding object. This ExternalInterface shall be derived from the AML standard interface class "COLLADAInterface" or one of its derivates.

  NOTE 2 The standard interface class "COLLADAInterface" is specified in 6.3.7. Details are intended to be specified in IEC 62714-3.

- PLCopen XML documents may be interlinked by utilizing corresponding PLCopen XML interfaces. If PLCopen XML items are required to be interlinked in CAEX, they shall be published by adding a CAEX ExternalInterface to the corresponding object. This ExternalInterface shall be derived from the AML standard interface class "PLCopenXMLInterface" or one of its derivates.

  NOTE 3 The standard interface class "PLCopenXMLInterface" is specified in 6.3.8. Details are intended to be specified in IEC 62714-4.

Figure 10a) describes an example comprising a robot "Rob1" and a PLC "PLC1", each with one signal interface that are connected. Figure 10b) depicts this example as an object hierarchy.

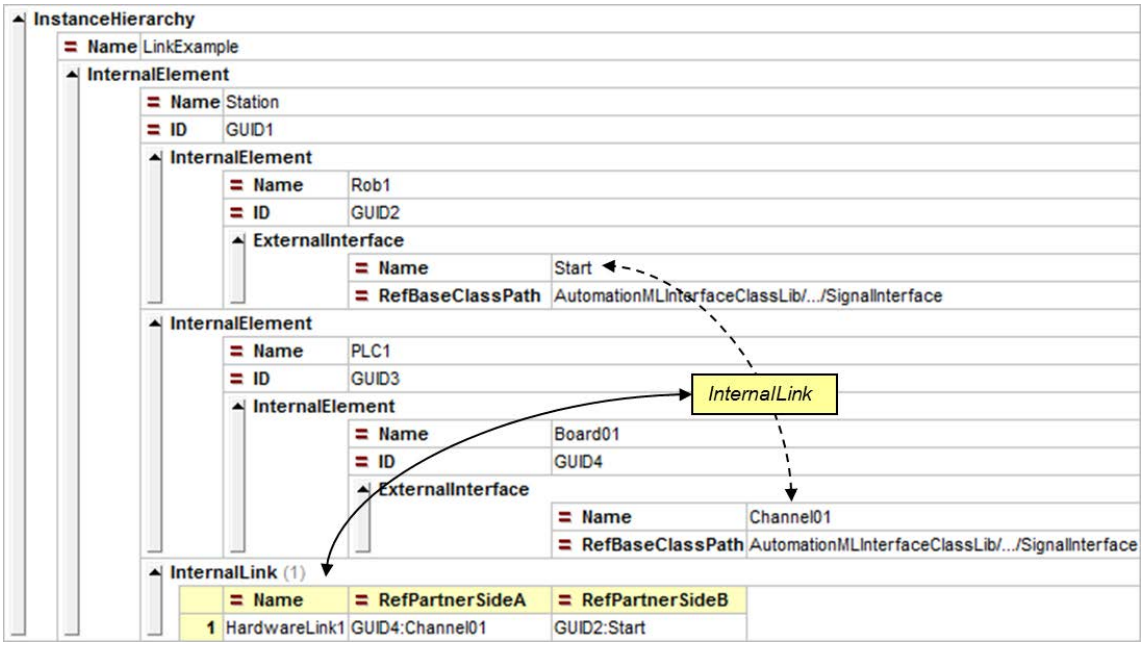


a) Relation as a block diagram **b) Relation as an object tree**

Figure 10 – Example of a relation as block diagram and as object tree

Figure 11 depicts the AML representation of the given example. The full XML text for the InstanceHierarchy for this example comprising all AML objects "Station", "Rob1", "PLC1" and "Board01" including their interfaces is shown below.

NOTE 4 The path strings given in this example are reduced with "/.../" in order to increase the readability.




```
<InstanceHierarchy Name="LinkExample">
  <InternalElement Name="Station" ID="GUID1">
    <InternalElement Name="Rob1" ID="GUID2">
      <ExternalInterface Name="Start" RefBaseClassPath="AutomationMLInterfaceClassLib/.../ SignalInterface">
        <Attribute Name="Type">
          <Value>digital</Value>
        </Attribute>
        <Attribute Name="Direction">
          <Value>In</Value>
        </Attribute>
      </ExternalInterface>
    </InternalElement>
    <InternalElement Name="PLC1" ID="GUID3">
      <InternalElement Name="Board01" ID="GUID4">
        <ExternalInterface Name="Channel01" RefBaseClassPath="AutomationMLInterfaceClassLib/.../ SignalInterface">
          <Attribute Name="Type">
            <Value>digital</Value>
          </Attribute>
          <Attribute Name="Direction">
            <Value>Out</Value>
          </Attribute>
        </ExternalInterface>
      </InternalElement>
    </InternalElement>
    <InternalLink Name="HardwareLink1" RefPartnerSideA="GUID4:Channel01" RefPartnerSideB="GUID2:Start"/>
  </InternalElement>
</InstanceHierarchy>
```

```
<InstanceHierarchy Name="LinkExample">
  <InternalElement Name="Station" ID="GUID1">
    <InternalElement Name="Rob1" ID="GUID2">
      <ExternalInterface Name="Start" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface"/>
    </InternalElement>
    <InternalElement Name="PLC1" ID="GUID3">
      <InternalElement Name="Board01" ID="GUID4">
        <ExternalInterface Name="Channel01" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterfa
      </InternalElement>
    </InternalElement>
    <InternalLink Name="HardwareLink1" RefPartnerSideA="GUID4:Channel01" RefPartnerSideB="GUID2:Start"/>
  </InternalElement>
</InstanceHierarchy>
```

Figure 11 – Example relation between the objects "PLC1" and "Rob1"

5.7 AML document reference specification

5.7.1 General

A document reference serves for the linking between one AML object and one external document which may contain e.g. geometry, kinematics or sequence information. The reference mechanism is based on the standard AML interface “ExternalDataConnector” or one of its derivatives.

5.7.2 Referencing COLLADA documents

Referencing COLLADA documents shall be done based on the AML standard interface class “COLLADAInterface” or one of its derivates. This class is specified in 6.3.7. Details are intended to be specified in IEC 62714-3.

5.7.3 Referencing PLCopen XML documents

Referencing PLCopen XML shall be done based on the AML standard interface “PLCopenXMLInterface” or one of its derivates. This class is specified in 6.3.8. Details are intended to be specified in IEC 62714-4.

5.7.4 Referencing additional documents

Future extensions of IEC 62714 may add additional interface types for referencing additional document types. They are specified in separate parts of IEC 62714 and not in the scope of this standard. For these extensions, the following provisions apply:

- If additional document types have to be added to IEC 62714, they shall be modelled with additional interface classes.
- These additional interfaces shall be modelled as extension of the AML InterfaceClass library and shall be directly or indirectly derived from the standard interface class ExternalDataConnector.
- The storage of references should be done using the same standard attributes provided by the standard interface classes.
- The standard interface class “ExternalDataConnector” shall only be used for document types included in IEC 62714.

6 AML base libraries

6.1 General

Clause 6 defines essential AML base libraries with AML base classes needed for the modelling of core AML concepts. All described attributes are part of the AML standard library and may be removed in the InstanceHierarchy if not needed.

NOTE Domain specific libraries are within the scope of further parts of IEC 62714.

6.2 General provisions

Regarding AML base libraries, the following provisions apply:

- All AML objects shall be associated directly or indirectly to the role class “AutomationMLBaseRole”.
- All interfaces shall be directly or indirectly associated to an AML interface class.
- AML attributes shall be used if required and may be removed from AML objects if not needed.

6.3 AML interface class library – AutomationMLInterfaceClassLib

6.3.1 General

The following AutomationMLInterfaceClassLib is modelled according to IEC 62424:2008, Clause 7, Annex A and Annex C. IEC 62714 utilizes the CAEX interface concept. User-defined extensions of this AML library are allowed as specified in 7.3.

Each interface shall be derived directly or indirectly from a class of the following standard AutomationMLInterfaceClassLib according to Table 3. Subclauses 6.3.2 to 6.3.10 specify the interface classes in detail.

Table 3 – Interface classes of the AutomationMLInterfaceClassLib

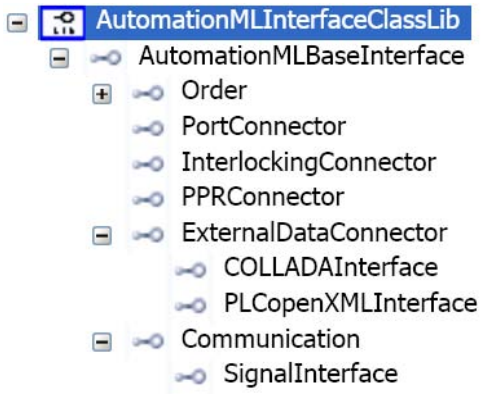
AML InterfaceClass library	InterfaceClass	Description
	AutomationMLBaseInterface	Abstract interface type
	Order	Interface for describing orders
	PortConnector	Interface for describing ports
	PPRConnector	Connector for interlinking products, resources or processes
	ExternalDataConnector	Generic connector interface to external data
	COLLADAInterface	Interface to a COLLADA document
	PLCopenXMLInterface	Interface to a PLCopen XML document
	Communication	Generic communication interface
	SignalInterface	Generic signal interface

Figure 12 shows a table view and Figure 13 the XML text of the standard AML InterfaceClassLib. Subclauses 6.3.2 to 6.3.10 provide detail information about the classes.

- **InterfaceClassLib**
  - **Name**: AutomationMLInterfaceClassLib
  - **Description**: Standard AutomationML Interface Class Library
  - **Version**: 2.1.1
  - **InterfaceClass**
    - **Name**: AutomationMLBaseInterface
    - **InterfaceClass**
      - **Name**: Order
      - **RefBaseClassPath**: AutomationMLBaseInterface
      - **Attribute**
        - **Name**: Direction
        - **AttributeDataType**: xs:string
    - **InterfaceClass**
      - **Name**: PortConnector
      - **RefBaseClassPath**: AutomationMLBaseInterface
    - **InterfaceClass**
      - **Name**: InterlockingConnector
      - **RefBaseClassPath**: AutomationMLBaseInterface
    - **InterfaceClass**
      - **Name**: PPRConnector
      - **RefBaseClassPath**: AutomationMLBaseInterface
    - **InterfaceClass**
      - **Name**: ExternalDataConnector
      - **RefBaseClassPath**: AutomationMLBaseInterface
      - **Attribute**
        - **Name**: refURI
        - **AttributeDataType**: xs:anyURI
      - **InterfaceClass** (2)

	Name	RefBaseClassPath
1	COLLADAInterface	ExternalDataConnector
2	PLCopenXMLInterface	ExternalDataConnector

    - **InterfaceClass**
      - **Name**: Communication
      - **RefBaseClassPath**: AutomationMLBaseInterface
      - **InterfaceClass** (1)

	Name	RefBaseClassPath
1	SignalInterface	Communication

Figure 12 – AML basic interface class library

```
<InterfaceClassLib Name="AutomationMLInterfaceClassLib">
  <Description>Standard AutomationML Interface Class Library</Description>
  <Version>2.1.1</Version>
  <InterfaceClass Name="AutomationMLBaseInterface">
    <InterfaceClass Name="Order" RefBaseClassPath="AutomationMLBaseInterface">
      <Attribute Name="Direction" AttributeDataType="xs:string"/>
    </InterfaceClass>
    <InterfaceClass Name="PortConnector" RefBaseClassPath="AutomationMLBaseInterface"/>
    <InterfaceClass Name="InterlockingConnector" RefBaseClassPath="AutomationMLBaseInterface"/>
    <InterfaceClass Name="PPRConnector" RefBaseClassPath="AutomationMLBaseInterface"/>
    <InterfaceClass Name="ExternalDataConnector" RefBaseClassPath="AutomationMLBaseInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI"/>
      <InterfaceClass Name="COLLADAInterface" RefBaseClassPath="ExternalDataConnector"/>
      <InterfaceClass Name="PLCopenXMLInterface" RefBaseClassPath="ExternalDataConnector"/>
    </InterfaceClass>
    <InterfaceClass Name="Communication" RefBaseClassPath="AutomationMLBaseInterface">
      <InterfaceClass Name="SignalInterface" RefBaseClassPath="Communication"/>
    </InterfaceClass>
  </InterfaceClass>
</InterfaceClassLib>
```

Figure 13 – XML description of the AML basic interface class library

6.3.2 InterfaceClass AutomationMLBaseInterface

Table 4 specifies the interface class “AutomationMLBaseInterface”.

Table 4 – InterfaceClass AutomationMLBaseInterface

Class name	AutomationMLBaseInterface	
Description	The interface class “AutomationMLBaseInterface” is a basic abstract interface type and shall be used as parent for the description of all AML interface classes.	
Parent class	None	
Attributes	None	

6.3.3 InterfaceClass Order

Table 5 specifies the interface class “Order”.

Table 5 – InterfaceClass Order

Class name	Order	
Description	The interface class “Order” is an abstract class that shall be used for the description of orders, e.g. a successor or a predecessor.	
Parent class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface	
Attributes	Direction (type=”xs:string”)	The attribute “Direction” shall be used in order to specify the direction. Permitted values are “In”, “Out” or “InOut”.

6.3.4 InterfaceClass PortConnector

Table 6 specifies the interface class “PortConnector”.

Table 6 – InterfaceClass PortConnector

Class name	PortConnector	
Description	The interface class “PortConnector” shall be used in order to provide a high level relation between ports. An overview of the Port concept is given in A.2.2.	
Parent class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface	
Attributes	None	

6.3.5 InterfaceClass PPRConnector

Table 7 specifies the interface class “PPRConnector”.

Table 7 – InterfaceClass PPRConnector

Class name	PPRConnector	
Description	The interface class “PPRConnector” shall be used in order to provide a relation between resources, products and processes. See A.2.6 for more information.	
Parent class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface	
Attributes	None	

6.3.6 InterfaceClass ExternalDataConnector

Table 8 specifies the interface class “ExternalDataConnector”.

Table 8 – InterfaceClass ExternalDataConnector

Class name	ExternalDataConnector	
Description	The interface class “ExternalDataConnector” is a basic abstract interface type and shall be used for the description of connector interfaces referencing external documents. The classes “COLLADAInterface” and “PLCopenXMLInterface” are derived from this class. All existing and future connector classes shall be derived directly or indirectly from this class.	
Parent class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface	
Attribute	refURI (type=”xs:anyURI”)	The attribute “refURI” shall be used in order to store the path to the reference external document.

6.3.7 InterfaceClass COLLADAInterface

Table 9 specifies the interface class “COLLADAInterface”. Details are intended to be specified in IEC 62714-3.

Table 9 – InterfaceClass COLLADAInterface

Class name	COLLADAInterface	
Description	The interface class “COLLADAInterface” shall be used in order to reference external COLLADA documents and to publish interfaces that are defined inside an external COLLADA document. Details are intended to be specified in IEC 62714-3.	
Parent class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector	
Attributes	None	

6.3.8 InterfaceClass PLCopenXMLInterface

Table 10 specifies the interface class “PLCopenXMLInterface”. Details are intended to be specified in IEC 62714-4.

Table 10 – InterfaceClass PLCopenXMLInterface

Class name	PLCopenXMLInterface	
Description	The interface class “PLCopenXMLInterface” shall be used in order to reference external PLCopen XML documents or to publish signals or variables that are defined inside of a PLCopen XML logic description. Details are intended to be specified in IEC 62714-4.	
Parent class	AutomationMLBaseInterface/ExternalDataConnector	
Attributes	None	

6.3.9 InterfaceClass Communication

Table 11 specifies the interface class “Communication”.

Table 11 – InterfaceClass Communication

Class name	Communication	
Description	The interface class “Communication” is an abstract interface type and shall be used for the description of communication related interfaces. Further communication related classes shall be directly or indirectly derived from this class.	
Parent class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface	
Attributes	None	

6.3.10 InterfaceClass SignalInterface

Table 12 specifies the interface class “SignalInterface”.

Table 12 – InterfaceClass SignalInterface

Class name	SignalInterface	
Description	The interface class “SignalInterface” shall be used for modelling signals. This interface type is configurable and allows description of digital and analog inputs and outputs as well as configurable inputS-outputs. An example is described in Figure 10.	
Parent class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication	
Attributes	None	

6.4 AML basic role class library – AutomationMLBaseRoleClassLib

6.4.1 General

Subclause 6.4 defines an AML base library of essential standard role classes required for the modelling of core AML concepts. A role is a class that describes an abstract functionality without defining the underlying technical implementation. Example role classes are a “Resource” or a “Robot”. While associating a role class to an AML object, this AML object gets a semantic. Additional extended libraries are intended to be described in IEC 62714-2. All described attributes are part of the AML standard library and may be removed in the InstanceHierarchy if not needed.

Each AML object and each user defined role class shall have a direct or indirect reference to one of the roles in this AML library. If a certain role is too specific, the next parent should be referenced. Figure 14 to 16 present the standard basic RoleClass as object tree, as XML table and as XML text. Details of each role class are given in 6.4.2 to 6.4.13.

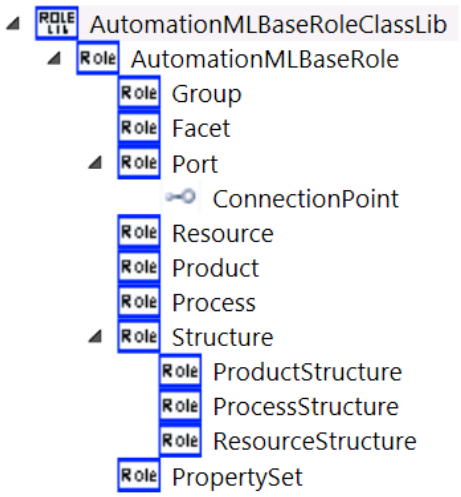


Figure 14 – AML basic role class library

RoleClassLib

Name	AutomationMLBaseRoleClassLib
Description	AutomationML base role library
Version	2.1.1

RoleClass

Name	AutomationMLBaseRole

RoleClass

Name	Group
RefBaseClassPath	AutomationMLBaseRole

Attribute (1)

	Name	AttributeDataType
1	AssociatedFacet	xs:string

RoleClass

Name	Facet
RefBaseClassPath	AutomationMLBaseRole

RoleClass

Name	Port
RefBaseClassPath	AutomationMLBaseRole

Attribute (3)

	Name	AttributeDataType	Attribute
1	Direction	xs:string	
2	Cardinality	xs:complexType	Attribute (2)
3	Category	xs:string	

Attribute (2)

	Name	AttributeDataType
1	MinOccur	xs:uint
2	MaxOccur	xs:uint

ExternalInterface

Name	ConnectionPoint
RefBaseClassPath	AutomationMLInterfaceClassLib@AutomationMLInterfaceClassLib/AutomationMLBaseInterface/PortConnector

RoleClass

Name	Resource
RefBaseClassPath	AutomationMLBaseRole

RoleClass

Name	Product
RefBaseClassPath	AutomationMLBaseRole

RoleClass

Name	Process
RefBaseClassPath	AutomationMLBaseRole

RoleClass

Name	Structure
RefBaseClassPath	AutomationMLBaseRole

RoleClass (3)

	Name	RefBaseClassPath
1	ProductStructure	AutomationMLBaseRole/Structure
2	ProcessStructure	AutomationMLBaseRole/Structure
3	ResourceStructure	AutomationMLBaseRole/Structure

RoleClass

Name	PropertySet
RefBaseClassPath	AutomationMLBaseRole

Figure 15 – AutomationMLBaseRoleClassLib

```
<RoleClassLib Name="AutomationMLBaseRoleClassLib">
  <Description>AutomationML base role library </Description>
  <Version>2.1.1</Version>
  <RoleClass Name="AutomationMLBaseRole">
    <RoleClass Name="Group" RefBaseClassPath="AutomationMLBaseRole">
      <Attribute Name="AssociatedFacet" AttributeDataType="xs:string"/>
    </RoleClass>
    <RoleClass Name="Facet" RefBaseClassPath="AutomationMLBaseRole"/>
    <RoleClass Name="Port" RefBaseClassPath="AutomationMLBaseRole">
      <Attribute Name="Direction" AttributeDataType="xs:string"/>
      <Attribute Name="Cardinality" AttributeDataType="xs:complexType">
        <Attribute Name="MinOccur" AttributeDataType="xs:uint"/>
        <Attribute Name="MaxOccur" AttributeDataType="xs:uint"/>
      </Attribute>
      <Attribute Name="Category" AttributeDataType="xs:string"/>
      <ExternalInterface Name="ConnectionPoint" RefBaseClassPath=
       "AutomationMLInterfaceClassLib@AutomationMLInterfaceClassLib/AutomationMLBaseInterface/PortConnector"/>
    </RoleClass>
    <RoleClass Name="Resource" RefBaseClassPath="AutomationMLBaseRole"/>
    <RoleClass Name="Product" RefBaseClassPath="AutomationMLBaseRole"/>
    <RoleClass Name="Process" RefBaseClassPath="AutomationMLBaseRole"/>
    <RoleClass Name="Structure" RefBaseClassPath="AutomationMLBaseRole">
      <RoleClass Name="ProductStructure" RefBaseClassPath="AutomationMLBaseRole/Structure"/>
      <RoleClass Name="ProcessStructure" RefBaseClassPath="AutomationMLBaseRole/Structure"/>
      <RoleClass Name="ResourceStructure" RefBaseClassPath="AutomationMLBaseRole/Structure"/>
    </RoleClass>
    <RoleClass Name="PropertySet" RefBaseClassPath="AutomationMLBaseRole"/>
  </RoleClass>
</RoleClassLib>
```

Figure 16 – XML text of the AutomationMLBaseRoleClassLib

6.4.2 RoleClass AutomationMLBaseRole

Table 13 specifies the role class “AutomationMLBaseRole”.

Table 13 – RoleClass AutomationMLBaseRole

Class name	AutomationMLBaseRole	
Description	The role class “AutomationMLBaseRole” is a basic abstract role type and the base class for all standard or user-defined role classes.	
Parent class	None	
Attributes	None	

6.4.3 RoleClass Group

Table 14 specifies the role class “Group”.

Table 14 – RoleClass Group

Class name	Group	
Description	The role class “Group” is a role type for objects that serve for the grouping of mirror objects that belong together from a certain engineering perspective. AML Group objects shall reference this role. Details and examples are specified in 8.4.	
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Attributes	“AssociatedFacet” (type = “xs:string”)	The attribute “AssociatedFacet” shall be used for the definition of the name of the corresponding Facet. Example: AssociatedFacet = “PLCFacet”.

6.4.4 RoleClass Facet

Table 15 specifies the role class “Facet”.

Table 15 – RoleClass Facet

Class name	Facet	
Description	The role class “Facet” is a role type for objects that serve as sub-view on attributes or interfaces of an AML object. AML Facet objects shall reference this role. Details and examples are specified in 8.3.	
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Attributes	None	

6.4.5 RoleClass Port

Table 16 specifies the role class “Port”.

Table 16 – Optional attributes for AML Port objects

Class name	Port	
Description	The role class “Port” is a role type for objects that groups a number of interfaces and allows describing complex interfaces in this way. AML Port objects shall reference this role. Details and examples are specified in 8.2.	
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Attributes	Direction (type = “xs:string”)	This attribute shall be used to describe the direction of the Port. Values shall be one of the following: “In”, “Out” or “InOut”. Ports with the direction “In” can only be connected to ports with the direction “Out” or “InOut” and ports with the direction “Out” can only be connected with ports with the direction “In” or “InOut”. Ports with the direction “InOut” can be connected to Ports of arbitrary direction. Examples: Direction = “Out” (e.g. a plug) Direction = “In” (e.g. a socket) Direction = “InOut” This information can be used e.g. in order to prove the validity of a connection. NOTE The validity of those connections is outside the scope of IEC 62714, but is a tool functionality.
	„Cardinality“	This attribute is a complex attribute and shall not have a value. The corresponding sub-attributes are described in Table 17.
	“Category” (type = “xs:string”)	The category attribute describes the Port type. The value of this attribute is user-defined. Only ports with the same category value are allowed to be connected. Example: Category = “MaterialFlow”.

The attribute “Cardinality” has two sub-attributes described in Table 17.

Table 17 – Sub-attributes of the attribute “Cardinality”

Attribute	Type	Description	Example
“MinOccur”	xs:unsignedInt	The MinOccur value describes the minimum possible number of connections to or from this Port. The attribute shall have values greater than or equal to 0.	MinOccur = 1 This means that this Port should be connected with at minimum one other Port.
“MaxOccur”	xs:unsignedInt	The MaxOccur describes the maximum possible number of connections to or from this Port. The attribute shall have values greater than or equal to MinOccur, or 0 which means infinite.	MaxOccur = 3 This means that this Port can only be connected with a maximum of three other ports.

Additionally the AML Port object shall have a CAEX ExternalInterface derived from the AML InterfaceClass “PortConnector” (see Table 18).

NOTE This interface allows connecting the considered Port with a number of other ports on an abstract level without detailed description of the inner relations between the sub-interfaces (see Figure A.13).

Table 18 – Interface of the AML Port class

Interface	Type	Description	Example
The name is user-defined, e.g. “ConnectionPoint”	PortConnector	This CAEX Interface allows connecting this Port with a number of other ports on an abstract level. The internal relations between single Port interfaces are not described in this way.	See A.2.2.2.

6.4.6 RoleClass Resource

Table 19 specifies the role class “Resource”.

Table 19 – RoleClass Resource

Class name	Resource	
Description	The role class “Resource” is a basic abstract role type and the base class for all AML resource roles. It describes plants, equipment or other production resources. AML resource objects shall directly or indirectly reference this role. Examples are specified in A.2.6.	
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Attributes	None	

Additionally, if required, AML resource objects shall have a CAEX ExternalInterface “PPRConnector” to create relations to products and processes (see 6.3.5).

6.4.7 RoleClass Product

Table 20 specifies the role class “Product”.

Table 20 – RoleClass Product

Class name	Product	
Description	The role class “Product” is a basic abstract role type and the base class for all AML product roles. It describes products, product parts or product related materials that are processed in the described plant. AML product objects shall directly or indirectly reference this role. Examples are specified in A.2.6.	
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Attributes	None	

Additionally, if required, AML product objects shall have a CAEX ExternalInterface “PPRConnector” to create relations to resources and processes (see 6.3.5).

6.4.8 RoleClass Process

Table 21 specifies the role class “Process”.

Table 21 – RoleClass Process

Class name	Process	
Description	The role class “Process” is a basic abstract role type and the base class for all AML process roles. It describes production related processes. AML process objects shall directly or indirectly reference this role. Examples are specified in A.2.6.	
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Attributes	None	

Additionally, if required, AML process objects shall have a CAEX ExternalInterface “PPRConnector” to create relations to products and resources (see 6.3.5).

6.4.9 RoleClass Structure

Table 22 specifies the role class “Structure”.

Table 22 – RoleClass Structure

Class name	Structure	
Description	The role class “Structure” is a basic abstract role type for objects that serve as structure elements in the plant hierarchy, e.g. a folder, a site or a manufacturing line. AML structure objects shall directly or indirectly reference this role.	
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Attributes	None	

6.4.10 RoleClass ProductStructure

Table 23 specifies the role class “ProductStructure”.

Table 23 – RoleClass ProductStructure

Class name	ProductStructure	
Description	The role class “ProductStructure” is an abstract role type for a product oriented object hierarchy. AML product structure objects shall directly or indirectly reference this role.	
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure	
Attributes	None	

6.4.11 RoleClass ProcessStructure

Table 24 specifies the role class “ProcessStructure”.

Table 24 – RoleClass ProcessStructure

Class name	ProcessStructure	
Description	The role class “ProcessStructure” is an abstract role type for a process oriented object hierarchy. AML process structure objects shall directly or indirectly reference this role.	
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure	
Attributes	None	

6.4.12 RoleClass ResourceStructure

Table 25 specifies the role class “ResourceStructure”.

Table 25 – RoleClass ResourceStructure

Class name	ResourceStructure	
Description	The role class “ResourceStructure” is an abstract role type for a resource oriented object hierarchy. AML resource structure objects shall directly or indirectly reference this role.	
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure	
Attributes	None	

6.4.13 RoleClass PropertySet

Table 26 specifies the role class “PropertySet”.

Table 26 – RoleClass PropertySet

Class name	PropertySet	
Description	The role class “PropertySet” is an abstract role type that serves for the definition of sets of properties corresponding to a certain engineering aspect. AML property set objects shall directly or indirectly reference this role. Normative provisions are described in 8.5, details and examples are specified in A.2.5.	
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Attributes	None	

7 Modelling of user-defined data

7.1 General

Clause 7 describes how user-defined data may be modelled in AML. Modelling of specific user-defined data is a core concept of AML. User-defined data are those CAEX Attributes, InterfaceClasses and RoleClasses which are not predefined by IEC 62714. The AML top-level data format CAEX provides mechanisms for modelling of user-defined data.

In order to allow the exchange of user-defined data, user specific agreements and functionality might therefore be required which are not part of IEC 62714. Source engineering tool specific meta information described in 5.4 supports those functionalities.

AML allows defining a relation between user-defined data and standard data by means of the Role Concept, the PropertySet Concept or standard CAEX mappings. These concepts ease the automatic interpretation of user-defined classes and attributes.

7.2 User-defined attributes

All attributes defined in IEC 62714 are called AML attributes. All attributes which are not defined in IEC 62714 are called user-defined attributes. AML attributes and user-defined attributes are stored in the same way as CAEX Attributes.

Regarding user-defined attributes, the following provisions apply:

- CAEX Attributes shall be stored in AML according to the CAEX Attribute definition in IEC 62424:2008, A.2.4.
- If units are required, user-defined attributes shall base on the same unit system. This part of IEC 62714 does not define a unit system.

  It is suggested to use SI units according to ISO 80000-1. For units regarding information technology, it is suggested to use IEC 60027.

Figure 17 gives an example of a user-defined object “Object01” with a user-defined attribute “Length”.

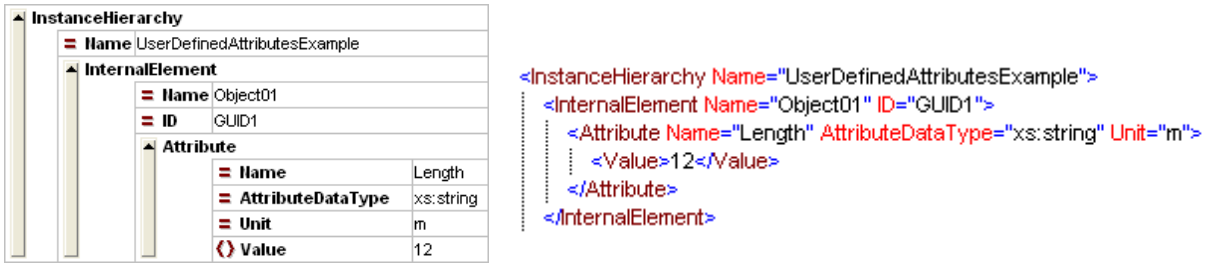


Figure 17 – Example of a user-defined attribute

7.3 User-defined InterfaceClasses

All InterfaceClasses defined in IEC 62714 are called AML InterfaceClasses. All InterfaceClasses not defined in IEC 62714 are called user-defined InterfaceClasses.

Regarding user-defined InterfaceClasses, the following provisions apply:

- All user-defined InterfaceClasses shall be stored according to the CAEX InterfaceClass definition in IEC 62424:2008, A.2.5.

  NOTE AML InterfaceClasses and user-defined InterfaceClasses are stored in the same way as CAEX InterfaceClasses.

- In order to ensure algorithmic interpretability of the semantic of user-defined InterfaceClasses, they shall be derived from AML InterfaceClasses.

Figure 18 shows an example of a user-defined class “MyDigitalInput” which is derived from the AML InterfaceClass “SignalInterface”. The inheritance relations between the InterfaceClass “MyDigitalInput” and the standard AML InterfaceClass “SignalInterface” allows the automatic identification of the user-defined class as a digital input interface. The user defined attributes are set properly. In this example, the user defined attributes are out of the scope of this part of IEC 62714.

NOTE This example uses a reduced notation of the path for increased readability. In real applications, the path is provided completely.

InterfaceClassLib

= Name	UserDefinedClassLib

InterfaceClass

= Name	MyDigitalInput
= RefBaseClassPath	AutomationMLInterfaceClassLib/.../ SignalInterface

Attribute (3)

	= Name	= AttributeDataType	{} Value
1	Type	xs:string	Digital
2	Direction	xs:string	In
3	Enabled	xs:boolean	true

```
<InterfaceClassLib Name="UserDefinedClassLib">
  <InterfaceClass Name="MyDigitalInput" RefBaseClassPath="AutomationMLInterfaceClassLib/.../ SignalInterface">
    <Attribute Name="Type" AttributeDataType="xs:string">
      <Value>Digital</Value>
    </Attribute>
    <Attribute Name="Direction" AttributeDataType="xs:string">
      <Value>In</Value>
    </Attribute>
    <Attribute Name="Enabled" AttributeDataType="xs:boolean">
      <Value>true</Value>
    </Attribute>
  </InterfaceClass>
</InterfaceClassLib>
```

Figure 18 – Example of a user-defined InterfaceClass in a user-defined InterfaceClassLib

7.4 User-defined RoleClasses

All RoleClasses defined in IEC 62714 are called AML RoleClasses. All RoleClasses not defined in IEC 62714 are called user-defined RoleClasses.

Regarding user-defined RoleClasses, the following provisions apply:

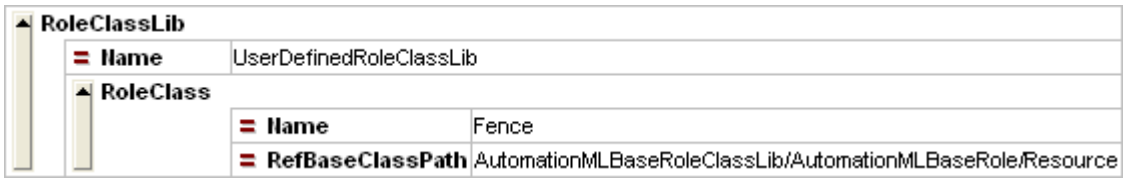
- CAEX RoleClasses shall be stored according to the CAEX RoleClass definition in IEC 62424:2008, A.2.6.

  NOTE 1 AML RoleClasses and user-defined RoleClasses are stored in the same way as CAEX RoleClasses.

- In order to ensure semantic interpretability of user-defined RoleClasses, they shall be derived from AML RoleClasses.

  NOTE 2 This serves for the algorithmic interpretability of the semantic of the class.

Figure 19 shows an example of a user-defined class “Fence” which is derived from the standard AML RoleClass “Resource”. The inheritance relation between “Fence” and “Resource” allows interpreting this user-defined class as a resource.




```
<RoleClassLib Name="UserDefinedRoleClassLib">
  <RoleClass Name="Fence" RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource"/>
</RoleClassLib>
```

Figure 19 – Example of a user-defined RoleClass in a user-defined RoleClassLib

7.5 User-defined SystemUnitClasses

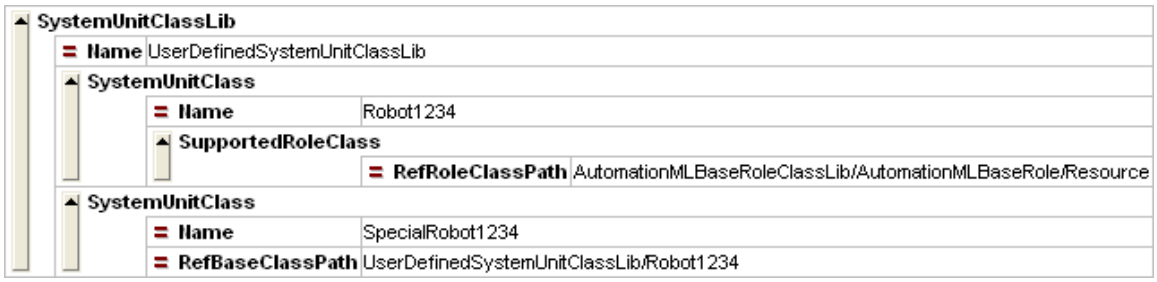
All SystemUnitClasses are user-defined. IEC 62714 does not specify SystemUnitClasses.

Regarding user-defined SystemUnitClasses, the following provisions apply:

- User-defined SystemUnitClasses shall be stored in AML according to the CAEX SystemUnitClass definition in IEC 62424:2008, A.2.3.
- User-defined SystemUnitClasses shall directly or indirectly be assigned to an AML RoleClass and shall use the AML attributes whenever applicable.

Examples: Figure 20 illustrates the definition of a user-defined SystemUnitClass by means of two different examples.

- The SystemUnitClass “Robot1234” depicts a user-defined class which supports the role “Resource” of the AML standard RoleClassLib. This class can therefore be automatically interpreted as “Resource”.
- The SystemUnitClass “SpecialRobot1234” depicts a new user-defined class which is derived from “Robot1234”. This class is therefore also a resource.




```
<SystemUnitClassLib Name="UserDefinedSystemUnitClassLib">
  <SystemUnitClass Name="Robot1234">
    <SupportedRoleClass RefRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource"/>
  </SystemUnitClass>
  <SystemUnitClass Name="SpecialRobot1234" RefBaseClassPath="UserDefinedSystemUnitClassLib/Robot1234"/>
</SystemUnitClassLib>
```

Figure 20 – Examples for different user-defined SystemUnitClasses

7.6 User-defined InstanceHierarchies

CAEX InstanceHierarchies serve for the storage of individual and project related engineering information. They form the centre of the AML top-level format and contain all individual data objects including properties, interfaces, relations and references.

Regarding user-defined InstanceHierarchies, the following provisions apply:

- This part of IEC 62714 does not restrict the depth of the hierarchy levels.
- This part of IEC 62714 does not restrict the architecture rules of a hierarchy.
- This part of IEC 62714 does not define naming conventions for the hierarchies.
- Every AML object within an InstanceHierarchy shall directly or indirectly be assigned to an AML RoleClass in order to specify its abstract type.

Figure 21 depicts an example project hierarchy that comprises a line “L001” with a station “S001” containing two robots “R0010_D” and “R0020_D” as well as a conveyor “RF010” and a PLC “P001”.

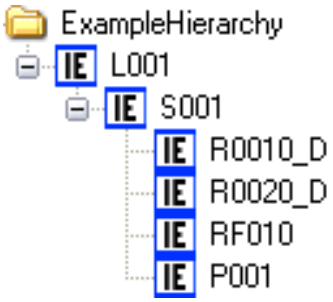


Figure 21 – Example of a user-defined InstanceHierarchy

Figure 22 shows the AML representation of this structure. According to IEC 62424:2008, A.2.9, every object has an association to a RoleClass.

```
<InstanceHierarchy Name="ExampleHierarchy">
  <InternalElement Name="L001" ID="GUID1">
    <InternalElement Name="S001" ID="GUID2">
      <InternalElement Name="R0010_D" ID="GUID3" RefBaseSystemUnitPath="RobotLibrary/Robot_1234">
        <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource"/>
      </InternalElement>
      <InternalElement Name="R0020_D" ID="GUID4" RefBaseSystemUnitPath="RobotLibrary/Robot_1234">
        <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource"/>
      </InternalElement>
      <InternalElement Name="RF010" ID="GUID5">
        <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource"/>
      </InternalElement>
      <InternalElement Name="P001" ID="GUID6">
        <RoleRequirements RefBaseRoleClassPath="AutomationMLCSRoleClassLib/ControlEquipment/ControlHardware/Controller/PLC"/>
      </InternalElement>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource/Structure"/>
    </InternalElement>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource/Structure"/>
  </InternalElement>
</InstanceHierarchy>
```

Figure 22 – AML representation of a user-defined InstanceHierarchy

8 Extended AML concepts

8.1 General overview

This part of IEC 62714 defines extended concepts for the modelling of specific engineering aspects. An informative overview and examples are provided in A.2.

8.2 AML Port object

An AML Port is an AML object that groups a number of interfaces. An informative overview about the Port concept including examples is provided in A.2.2.

Regarding AML Ports, the following provisions apply:

- An AML Port shall be described by a CAEX InternalElement with an association to the RoleClass “Port” which is described in 6.4.5.
- An AML Port object shall be modelled as a child object of the considered AML object or class.
- The required collection of interfaces shall be described by CAEX ExternalInterfaces of the Port object.
- A Port object shall not contain child CAEX InternalElements.
- All CAEX ExternalInterfaces of the Port object should directly or indirectly be derived from an AML interface class defined in 6.3.
- An AML Port shall additionally have at least one CAEX ExternalInterface which is derived from the AML InterfaceClass “PortConnector” described in 6.3.4.

NOTE Additional normative provisions regarding Port object attributes are provided in 6.4.5.

8.3 AML Facet object

A Facet is an AML object providing a sub-view on attributes or interfaces of the parent AML object. This concept serves for the storage of different configuration settings such as HMI or PLC related data and allows the automation of several control engineering steps. For this, this part of IEC 62714 defines the AML RoleClass “Facet" (see 6.4.4). An informative overview about the Facet concept including examples is provided in A.2.3.

Regarding AML Facets, the following provisions apply:

- An AML Facet object shall be described by a CAEX InternalElement with an association to the RoleClass “Facet” which is described in 6.4.4.
- An AML Facet object shall be modelled as a child object of the considered AML object or class.
- Facets shall have a unique arbitrary name among the siblings.

  NOTE The Facet name is important for the association with the Group concept. See A.2.3 for a concept description and examples.
- An AML object or class may have an arbitrary number of Facet objects.
- Facets may have an arbitrary number of Facet attributes.
- A Facet attribute shall be related to an existing attribute of the parent AML object, the identifier is the same name. Facet attributes which are not part of the parent object are not permitted.
- A Facet interface shall be related to an existing interface of the parent object, the identifier is the same name. Facet interfaces which are not part of the parent object are not permitted.
- Facets shall not contain new child objects, attributes or interfaces.
- Facet objects shall not be nested.
- Facets shall not modify existing attributes or interfaces.

8.4 AML Group object

The AML Group concept allows separating structure information from instance information. An informative overview about the Group concept including examples is provided in A.2.4.

Regarding AML Group objects, the following provisions apply:

- An AML Group object shall be described by a CAEX InternalElement with an association to the RoleClass “Group” which is defined in 6.4.3.

- An AML Group object may be modelled at an arbitrary position of the InstanceHierarchy or a SystemUnitClass.
- The number of AML Group objects is not limited.
- An AML Group object shall only contain mirror objects and/or further Group objects.

  NOTE 1 Thus, Group objects can be nested.

  NOTE 2 If an instance A references to another instance A*, A is called “mirror object” and A* is called “master object” (according to IEC 62424:2008, A.2.14). A mirror object references the master object and all data of it. Thus, a mirror object acts as a pointer to the master object.
- AML Groups shall not be used to describe plant hierarchies.
- An AML Group object may store additional information as attributes, interfaces or ports in order to describe group specific information.

  NOTE 3 Those additional attributes, ports and interfaces are not identical to attributes, ports or interfaces of the contained mirror objects.
- It is not allowed to change existing attributes, interfaces or ports of mirror objects or to add additional information to the mirror objects.
- A mirror object shall have an own unique ID.

  NOTE 4 A mirror object is considered to be identical to the master object. The ID supports distinguishing the mirror representation from the master.
- If a master object is deleted, all corresponding mirror objects shall be deleted too in order to avoid inconsistencies.

  NOTE 5 This is a tool functionality which is out of the scope of this part of IEC 62714.
- If a mirror object is deleted, the master object shall not be affected.
- If used, the attribute “AssociatedFacet” shall have a value that provides a valid name of an existing Facet.

8.5 AML PropertySet

A PropertySet is a role class containing a set of attributes with a well-defined syntax and semantic. It is modelled as role class derived from the standard role class “PropertySet”. A.2.5 gives a conceptual overview.

Regarding the PropertySet concept, the following provisions apply:

- A PropertySet class shall be modelled as role class and shall be directly or indirectly derived from the standard role class “PropertySet”.
- PropertySet classes may be collected in one or multiple role class libraries.
- AML objects may be associated with one or more property-set-classes.
- For each PropertySet of an AML object, a separate child CAEX InternalElement of the AML object shall be created which shall not define any CAEX attributes, interfaces or InternalElements except a name and an ID. This child object shall associate the intended PropertySet role class by means of the CAEX element “RoleRequirement”.
- Mappings between attributes of the AML object and a PropertySet role shall be modelled by means of the CAEX elements “MappingObject” and “AttributeNameMapping” within the corresponding child InternalElement. These mappings are valid between the belonging AML object and the referenced PropertySet. Mapped attributes shall be copied to the RoleRequirement section. No mapped attributes may be copied into the RoleRequirement section.
- Attributes of a PropertySet may be nested.
- Associations between an AML object and multiple property sets shall be modelled by means of multiple child elements of the AML object with each its own RoleRequirement association to the corresponding property set and each its own mappings.

Figure 23 illustrates this by means of an example. The object Robot_1 has a number of user-defined attributes. A child InternalElement IE is associated with the PropertySet “Geometry”

which defines the attributes. The MappingObject of IE specifies the mapping of the proprietary and the standardized attributes.

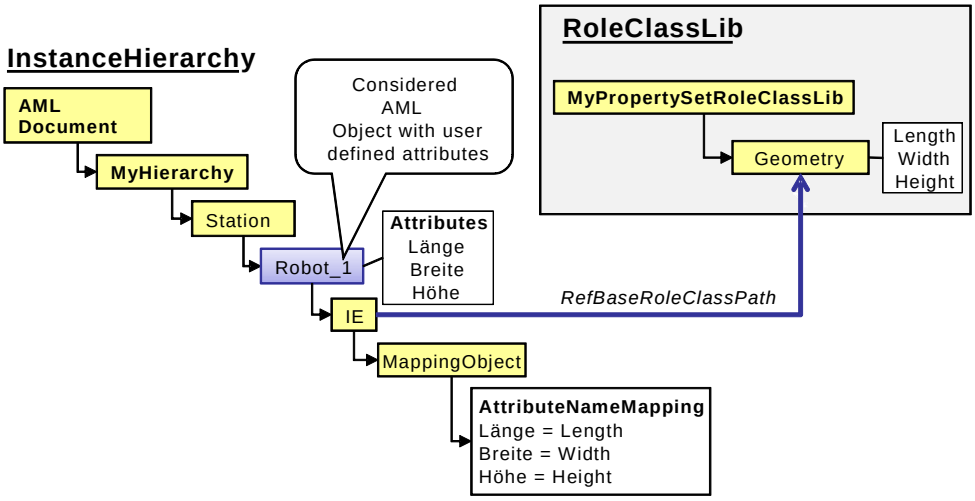
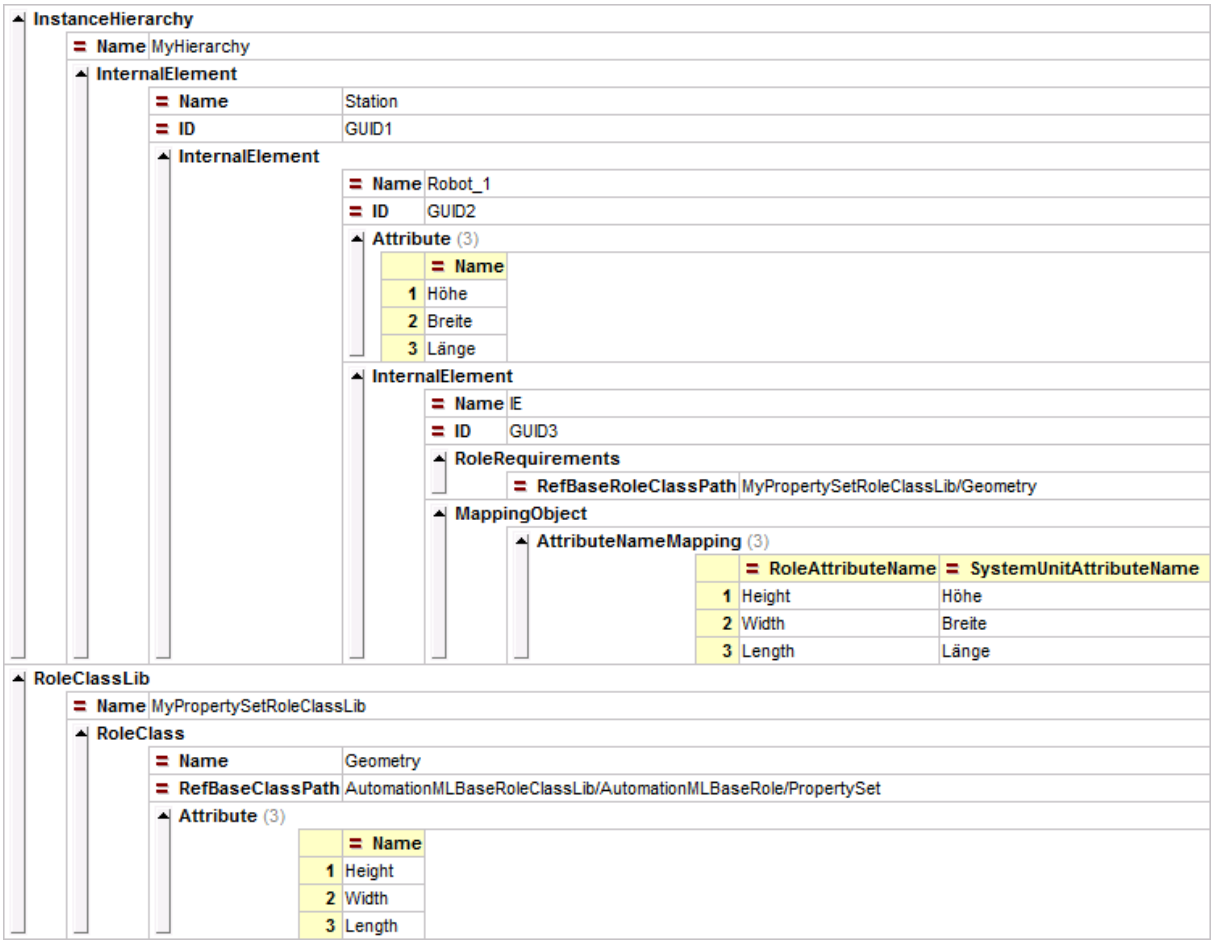


Figure 23 – Example illustrating the PropertySet concept




```
<InstanceHierarchy Name="MyHierarchy">
   <InternalElement Name="Station" ID="GUID1">
      <InternalElement Name="Robot_1" ID="GUID2">
         <Attribute Name="Höhe"/>
         <Attribute Name="Breite"/>
         <Attribute Name="Länge"/>
         <InternalElement Name="IE" ID="GUID3">
            <RoleRequirements RefBaseRoleClassPath="MyPropertySetRoleClassLib/Geometry"/>
            <MappingObject>
               <AttributeNameMapping RoleAttributeName="Height" SystemUnitAttributeName="Höhe"/>
               <AttributeNameMapping RoleAttributeName="Width" SystemUnitAttributeName="Breite"/>
               <AttributeNameMapping RoleAttributeName="Length" SystemUnitAttributeName="Länge"/>
            </MappingObject>
         </InternalElement>
      </InternalElement>
   </InternalElement>
</InstanceHierarchy>
<RoleClassLib Name="MyPropertySetRoleClassLib">
   <RoleClass Name="Geometry" RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/PropertySet">
      <Attribute Name="Height"/>
      <Attribute Name="Width"/>
      <Attribute Name="Length"/>
   </RoleClass>
</RoleClassLib>
```

Figure 24 – XML text of the PropertySet example

8.6 Support of multiple roles

In addition to IEC 62424:2008, A.3.18, this part of IEC 62714 defines how to specify multiple roles support for an object instance. Multiple roles are of interest, if an object can have

multiple functionalities. An example is a device that is a scanner, a printer or a fax device at the same time. Subclause A.2.7 gives an overview and describes a corresponding example.

Regarding the support of multiple roles, the following provisions apply:

- If an instance supports only one role, the corresponding role shall be specified using the CAEX attribute “RefBaseRoleClassPath” of the belonging RoleRequirement.

  NOTE 1 This is according to IEC 62424:2008, A.3.18, which only defines the support of one role at the same time.

- If an instance supports multiple roles, they shall be defined using each a CAEX element “SupportedRoleClass” instead of the CAEX attribute “RefBaseRoleClassPath”.

  NOTE 2 The attribute “RefBaseRoleClassPath” can only be assigned one time at the RoleRequirement element whereas the CAEX element “SupportedRoleClass” can be defined multiple times. This is the key to assigning multiple roles. However, this is a slight semantic extension according to IEC 62424 but does not change the CAEX data format.

- If an instance supports multiple roles and the requirements to the different roles shall be stored at the instance, this shall be done using the CAEX element “RoleRequirements” whereas the corresponding attributes or interfaces are directly assigned including the role name, a separator string “.” and the attribute or interface name.

  NOTE 3 This is a slight semantic extension according to IEC 62424 but does not change the CAEX data format. The difference to IEC 62424:2008, A.3.18, is that the role name is added to the attribute or interface definition. An example is provided in A.2.7.

- If several supported role classes are specified and the CAEX element “RoleRequirements” associates a certain “RefBaseRoleClassPath” at the same time, then the associated role class is the preferred role. In this case, RoleRequirements attribute definitions and attribute or interface mappings without an explicit role name prefix are associated with the preferred role.

  NOTE 4 For this preferred role, the usage is according to IEC 62424:2008, Annex A, without semantic extension.

8.7 Splitting of AML top-level data into different documents

According to IEC 62424:2008, A.2.12, CAEX explicitly supports the distribution of engineering data into different files and provides mechanisms to reference external CAEX files by means of the CAEX element “ExternalReference” and the corresponding Alias-Concept of CAEX.

8.8 Internationalization

Different languages for e.g. names and descriptions may be stored in AML in conformity with the XML specifications based on UTF-8.

8.9 Version information of AML objects

For the storage of version and revision information of individual AML objects (object instances) the standard version and revision fields according to IEC 62424:2008, A.2.2.2, shall be used.

For the storage of AML related version information and AML library related version information, see 5.3.

For the storage of tool specific meta information, see 5.4.

Annex A
(informative)

General introduction into the Automation Markup Language

A.1 General Automation Markup Language concepts

A.1.1 The Automation Markup Language architecture

The Automation Markup Language is an XML schema-based data format designed for the vendor independent exchange of plant engineering information. The goal of AML is to inter-connect engineering tools from the existing heterogeneous tool landscape in their different disciplines, e.g. mechanical plant engineering, electrical design, process engineering, process control engineering, HMI development, PLC programming, robot programming, etc.

AML stores engineering information following the object oriented paradigm and allows model-ling of real plant components as data objects encapsulating different aspects. An object may consist of other sub-objects, and may itself be part of a larger composition or aggregation. It may describe e.g. a signal, a PLC, a tank, a control valve, a robot, a manufacturing cell in different levels of detail or a complete site, line or plant. Typical objects in plant automation comprise information on topology, geometry, kinematics and logic, whereas logic comprises sequencing, behaviour and control.

AML combines existing industry data formats that are designed for the storage and exchange of different aspects of engineering information. These data formats are used on an “as-is” basis within their own specifications and are not branched for AML needs.

The core of AML is the top-level data format CAEX that connects the different data formats. Therefore, AML has an inherent distributed document architecture.

Figure A.1 illustrates the basic AML architecture and the distribution of topology, geometry, kinematics and logic information.

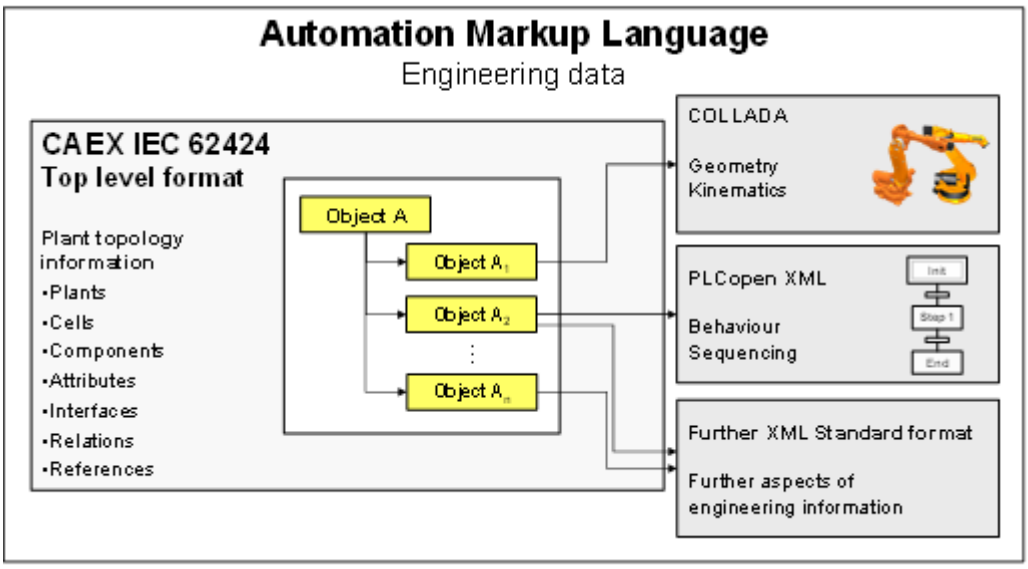


Figure A.1 – AML general architecture

The main advantages of the distributed document concept are the usage of proven and established data formats, the distribution of data to different files which eases the handling of bulk information and the simplified usage of AML library files which may be stored, exchanged and accessed separately. Finally, different levels of detail, e.g. geometry variants, may be stored separately. AML mainly defines the associations between the referenced data formats and engineering objects.

Using brief examples, A.1.1 gives a general overview about the information that may be stored and exchanged with AML.

- **Plant topology information**: The plant topology describes a plant as a hierarchical structure of individual plant objects which are represented by individual data objects. The object structure is modelled up to a certain level of detail (e.g. robot, gripper, but not axles or joints); the objects comprise properties and relations to other objects in their hierarchical structure. The plant topology acts as the top-level data structure and is stored by means of the CAEX data format according to IEC 62424:2008, Clause 7, Annex A and Annex C. In extension to IEC 62424, AML defines references from CAEX objects to information stored in external documents outside of CAEX. Subclause A.1.2 provides examples of how plant topology information is modelled with AML.
- **Geometry and kinematics information**: The geometry of a single plant object comprises its geometrical representation. The kinematics information describes the physical connections of 3D solids and the dependencies among objects. Both geometry and kinematics information are stored using the file format COLLADA". Additionally, the COLLADA file includes the definition of the coupling of geometry and kinematics information. COLLADA interfaces may be published as CAEX ExternalInterfaces within the top-level format for later interlinking. Out of the COLLADA geometry information of different objects, a complete scene may be derived automatically. These files may be referenced from CAEX and may be interlinked using CAEX linking mechanisms. Subclause A.1.3 provides a short example. Details are intended to be specified in IEC 62714-3.
- **Logic information**: The logic information describes sequences of actions and the behaviour of objects including I/O connections and logical variables. Sequences are described and stored in external PLCopen XML documents. Variables or signals may be published as CAEX ExternalInterfaces. These documents may be referenced out of CAEX and may be interlinked within CAEX. Subclause A.1.4 provides a short introduction about the main concepts. Details are intended to be specified in IEC 62714-4.
- **Reference and relation information**: AML distinguishes between references and relations. References depict links from CAEX objects to externally stored information. Relations depict associations between CAEX objects. Furthermore, the same mechanism is used in order to store associations between information stored in external documents. For this, it is necessary to publish the related link partners by means of CAEX ExternalInterfaces in the CAEX plant topology. Details about referencing COLLADA and PLCopen XML documents are intended to be provided in IEC 62714-3 and IEC 62714-4. Subclause A.1.5 provides an informative overview about the modelling of references and relations with AML. Subclauses 5.6 and 5.7 specify the normative provisions.
- **Referencing other data formats**: IEC 62714 may be extended in the future by additional parts specifying the integration of further data formats utilizing the AML reference mechanisms.

The exchange of engineering information additionally requires certain extended concepts. Clause A.2 explains these concepts and Clause 8 specifies their normative provisions.

NOTE In this document, paths are sometimes depicted in a reduced form, e.g. instead of "AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Port" they are noted in the form "AutomationMLInterfaceClassLib/.../Port". This serves the readability of the document. In real XML documents, all paths are stored according to this part of IEC 62714.

A.1.2 Modelling of plant topology information

In AML, real plant components are modelled as data objects encapsulating different aspects of engineering information. For this, it is necessary to structure the data objects. An established way to structure such data objects is an object hierarchy which is the plant topology (see 3.1.20).

In order to store hierarchical plant structures, AML utilizes concepts provided by the top-level data format CAEX according to IEC 62424:2008, A.2.11. Figure A.2 shows an example of a plant topology of a manufacturing line containing several objects of different hierarchical levels.

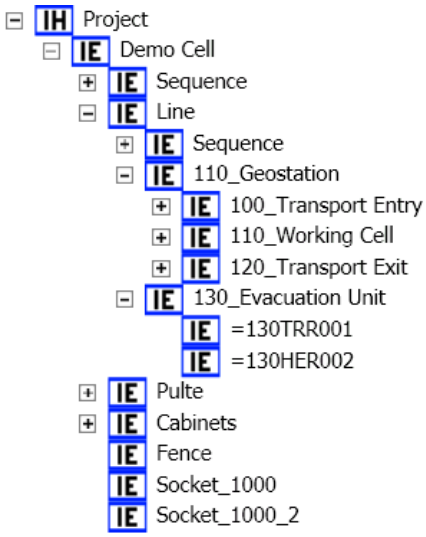


Figure A.2 – Plant topology with AML

Multiple hierarchies, crossed structures and complex object networks may be modelled using standard CAEX concepts. However, the key of the efficiency of the object oriented paradigm is the availability of libraries which contain predefined and proven solutions. For this, CAEX provides a number of different AML library types for interfaces, roles, system units and instance hierarchies which are of importance for AML.

- **InterfaceClasses and InterfaceClassLib:** Interfaces serve for the definition of relations between AML objects. Subclause 6.3 specifies the standard AML-InterfaceClassLib with a set of abstract InterfaceClasses which are dedicated to general automation systems. These classes comprise a syntactic and semantic definition and additionally serve the specification of user-defined object interfaces. Subclause 7.3 specifies the modelling of user-defined role classes.
- **RoleClasses and RoleClassLib:** RoleClasses serve for the definition of abstract characteristics of CAEX objects and thus serve the automatic semantic interpretation of user-defined AML objects. Subclause 6.4 specifies the basic AML-RoleClassLib with a set of abstract RoleClasses which are dedicated to general automation systems. Subclause 7.4 specifies the modelling of user-defined role classes. It is intended to specify further role libraries in IEC 62714-2.
- **SystemUnits and SystemUnitClassLib:** The SystemUnitClassLib is used to store vendor specific AML classes. Subclause 7.5 specifies architecture rules for the definition of SystemUnitClasses. AML does not predefine a certain SystemUnitClassLib or SystemUnitClass.
- **Instances and InstanceHierarchy:** Instance hierarchies store topology data of actual projects and are therefore the core of AML. They consist of AML object instances. Subclause 7.6 specifies how to store engineering information by means of instance hierarchies of type InstanceHierarchy.

An important aspect in the modelling of a plant topology is the identification of objects. Different engineering tools use different concepts for the identification of objects, e.g. a unique name, a unique identifier or a unique path. Some tools allow changes of the identifiers over the life time, others don't. Within one tool, this works fine, but exchanging the objects between different tools is not possible. For this, 5.5 specifies a mandatory object identification concept. Only such a concept enables the data exchange between different engineering tools with individual object identification concepts.

A.1.3 Referencing geometry and kinematics information

Geometry and kinematics information is stored in separate documents following the COLLADA data format. Modelling geometry and kinematics information is therefore split into two parts. On the one hand, the corresponding object is modelled within CAEX without any geometry or kinematics information as described in this part of IEC 62714. On the other hand, a COLLADA document has to be provided containing the geometry and kinematics information. Finally, the CAEX object stores a reference to the COLLADA document as is intended to be described in IEC 62714-3.

Figure A.3 shows an example AML document comprising the object "110RB_200", which references an external COLLADA document that contains the corresponding geometry and kinematics information.

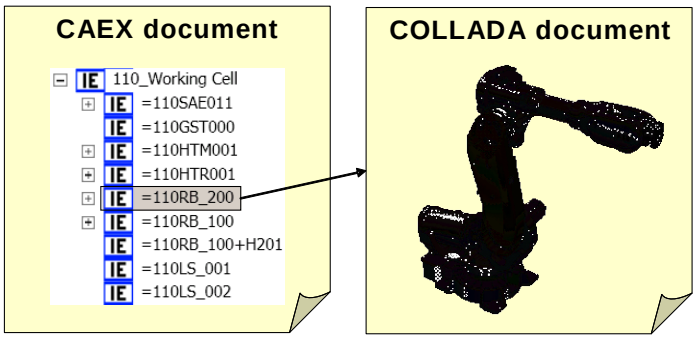


Figure A.3 – Reference from CAEX to a COLLADA document

The reference is modelled by means of a CAEX interface derived from the standard AML interface class "COLLADAInterface" specified in 5.7 and 6.3.7. Details are intended to be specified in IEC 62714-3.

A.1.4 Referencing logic information

Logics information is stored in separate documents following the PLCopen XML data format. Modelling logics information is therefore divided into two parts. On the one hand, the corresponding object is modelled within CAEX without any logics information as described in the present part of IEC 62714. On the other hand, a PLCopen XML document has to be provided containing the logics information as is intended to be described in IEC 62714-4. Finally, the CAEX object stores a reference to the PLCopen XML document. Figure A.4 shows an example AML document comprising the object "110_Working Cell", which references an external PLCopen XML document that contains the corresponding logic information.

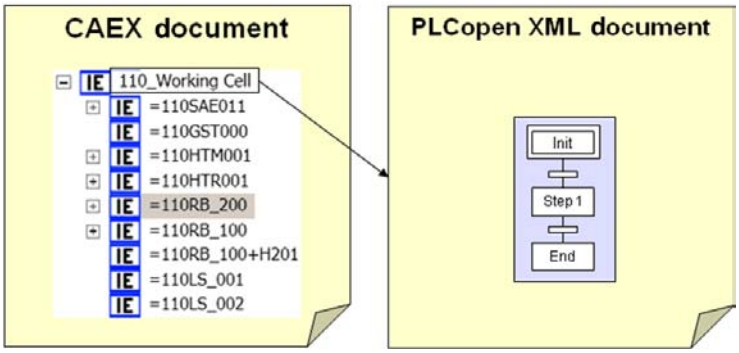


Figure A.4 – Reference from a CAEX to a PLCopen XML document

A.1.5 Modelling of relations

Modelling objects makes it necessary to define mechanisms to set these objects in relation to each other. Additional mechanisms are needed to link these objects with external stored data.

A relation expresses an association between two or more objects. This dependency may be of any nature including physical and logical dependencies. AML supports the following relations:

- parent-child-relations (see 5.6.2 and 5.6.3)
  - parent-child-relations between AML objects
  - parent-child-relations between AML classes
- inheritance relations (see 5.6.4)
  - inheritance relations between SystemUnitClasses
  - inheritance relations between RoleClasses
  - inheritance relations between InterfaceClasses
- class-instance-relations (see 5.6.5)
  - relations between a SystemUnitClass and an instance of it
  - relations between a RoleClass and an instance of it
  - relations between an InterfaceClass and an instance of it
- instance-instance-relations (see 5.6.6)
  - relations between AML objects
  - relations between published externally stored data

Figure A.5 presents the mentioned relation types supported by AML by means of an example.

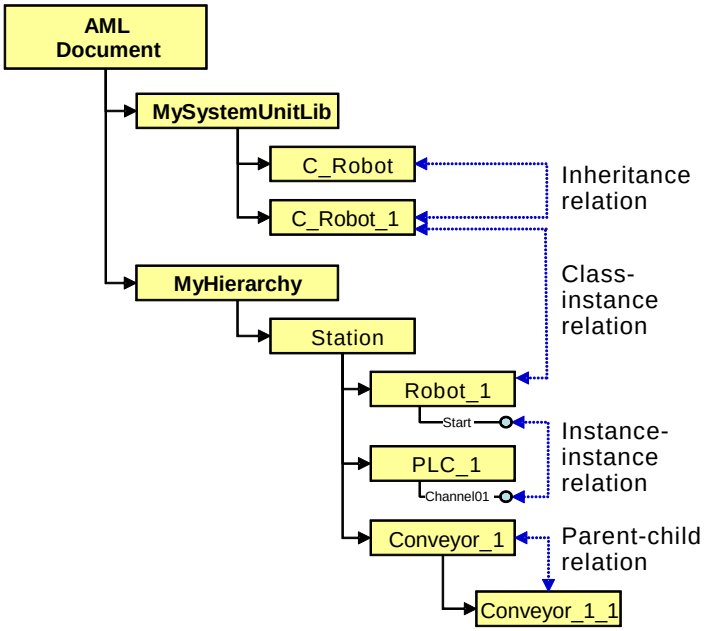


Figure A.5 – Relations in AML

Figure A.6 illustrates the AML model corresponding to the example by means of a table view. Note, that the path information is particularly reduced using the placeholder "/…/" in order to increase the readability. Figure A.7 shows the corresponding XML text of the AML library.

Figure A.8 shows the XML text of the InstanceHierarchy.

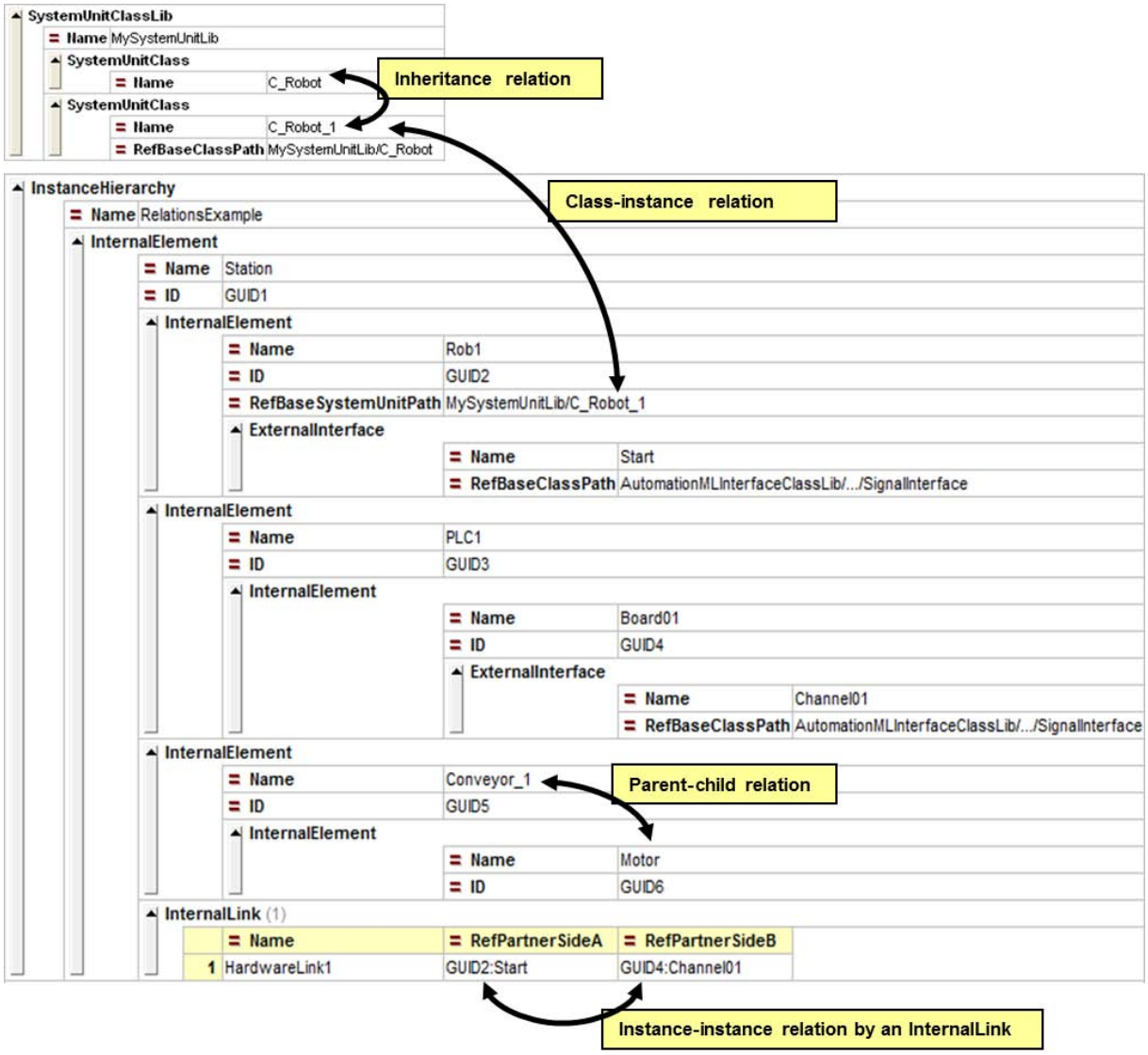


Figure A.6 – XML description of the relations example

```
<SystemUnitClassLib Name="MySystemUnitLib">
  <SystemUnitClass Name="C_Robot"/>
  <SystemUnitClass Name="C_Robot_1" RefBaseClassPath="MySystemUnitLib/C_Robot"/>
</SystemUnitClassLib>
```

Figure A.7 – XML text of the SystemUnitClassLib of the relations example

```
<InstanceHierarchy Name="RelationsExample">
  <InternalElement Name="Station" ID="GUID1">
    <InternalElement Name="Rob1" ID="GUID2" RefBaseSystemUnitPath="MySystemUnitLib/C_Robot_1">
      <ExternalInterface Name="Start" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface"/>
    </InternalElement>
    <InternalElement Name="PLC1" ID="GUID3">
      <InternalElement Name="Board01" ID="GUID4">
        <ExternalInterface Name="Channel01" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface"/>
      </InternalElement>
    </InternalElement>
    <InternalElement Name="Conveyor_1" ID="GUID5">
      <InternalElement Name="Motor" ID="GUID6"/>
    </InternalElement>
    <InternalLink Name="HardwareLink1" RefPartnerSideA="GUID2:Start" RefPartnerSideB="GUID4:Channel01"/>
  </InternalElement>
```

Figure A.8 – XML text of the InstanceHierarchy of the relations example

A.2 Extended AML concepts and examples

A.2.1 General overview

AML defines extended concepts for the modelling of specific engineering aspects such as the AML Port concept, the AML Facet concept and the AML Group concept. Table A.1 gives an overview of these concepts.

Table A.1 – Overview of major extended AML concepts

Concept	Description
AML Port	The Port concept allows a high level description of complex interfaces. AML Ports consist of a set of AML interfaces that belong together. They can be understood similar to plugs or sockets.
AML Facet	AML Facets allow the storage of a subset of attributes and interfaces of an AML object. They can be considered as views on engineering data.
AML Group	AML Groups allow the storage of separate views on a subset of AML objects. They can be used to filter objects of the plant tree for different engineering tools.
PropertySet	The PropertySet concept allows mapping proprietary attributes of user-defined AML objects with semantically predefined attributes. These semantically agreed attributes are stored in PropertySet role classes.
Process-Product-Resource	The Process-Product-Resource concept allows high level structuring of engineering data based on a process-centric, product-centric or resource-centric view including relations between them.

A.2.2 AML Port concept

A.2.2.1 Concept description

An AML Port is an AML object that groups a number of interfaces (see Figure A.9). A Port object belongs to one parent AML object and describes complex interfaces of the parent object. Ports may be connected to each other on a higher abstraction level instead of linking each single interface. AML Ports are useful in order to describe plugs, sockets or any other groups of interfaces which may directly be connected to each other. For this, AML defines the AML RoleClass “Port" (see 6.4.5). Normative provisions are specified in 8.2.

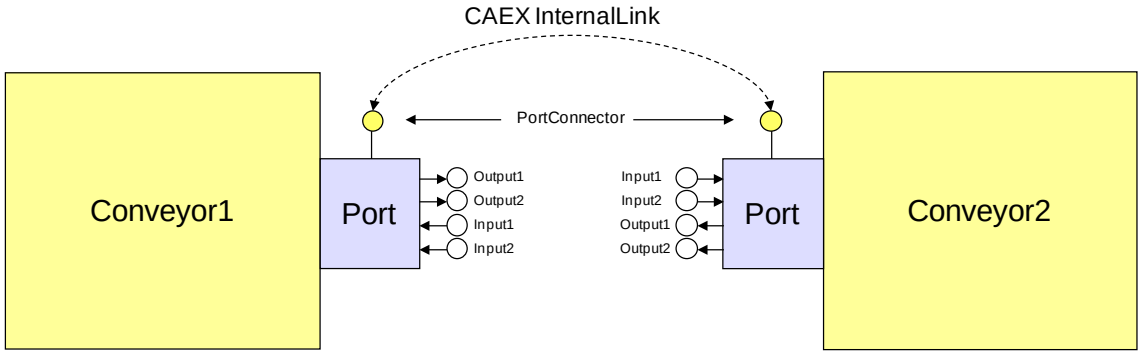


Figure A.9 – Port concept

A.2.2.2 Example

Figure A.10 gives an example for the AML Port concept. The object “Station” comprises the sub-objects “Conveyor1” and “Conveyor2”. Both sub-objects have each one Port object. The Port object comprises a collection of interfaces as well as a standard interface “Connection-Point” derived from the AML InterfaceClass “PortConnector”. This standard interface may be linked using a CAEX InternalLink. This relation means that both ports are connected to each other. The internal linking of the sub-interfaces is not described in detail, only the abstract ConnectionPoints are connected. In addition to this concept, AML allows storage of each indi-vidual link between the sub-interfaces.

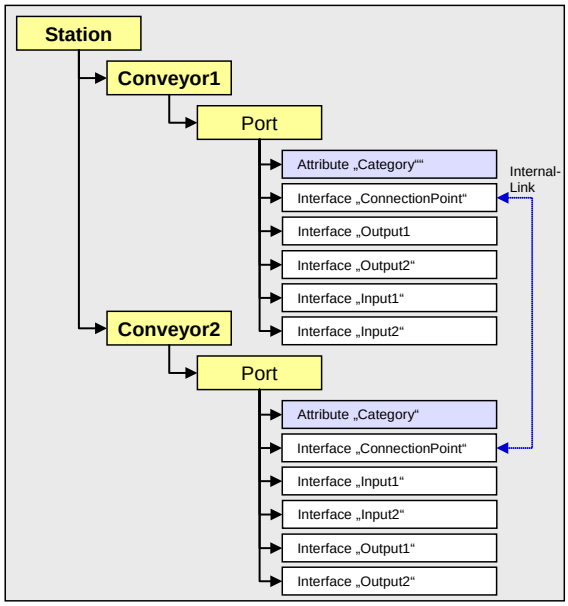
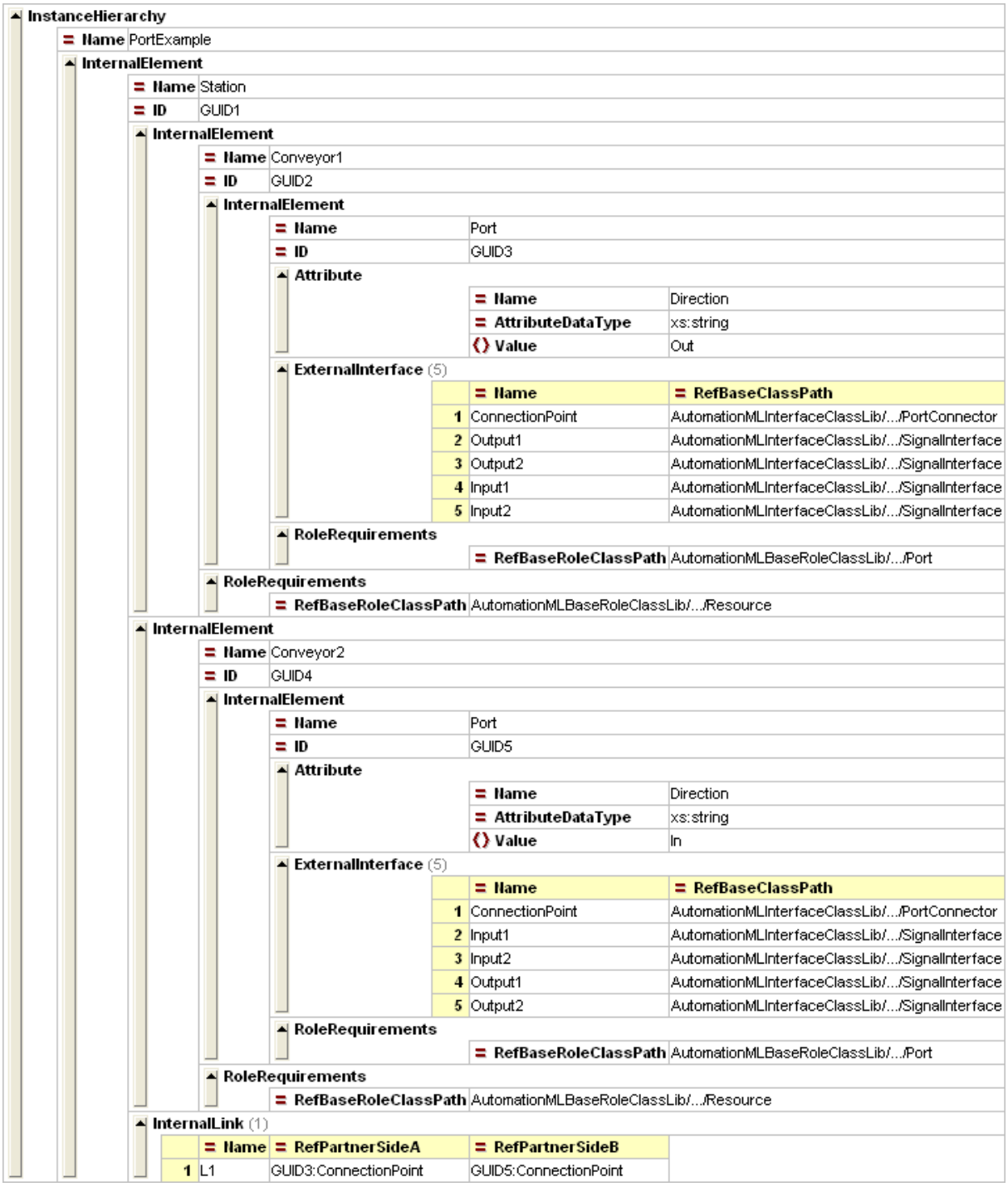


Figure A.10 – Example describing the AML Port concept

Figure A.11 and Figure A.12 describe the AML implementation of the example system described in Figure A.10.



- InstanceHierarchy
  - Name: PortExample
  - InternalElement
    - Name: Station
    - ID: GUID1
    - InternalElement
      - Name: Conveyor1
      - ID: GUID2
      - InternalElement
        - Name: Port
        - ID: GUID3
        - Attribute

Name	Value
Direction	Out
xs:string	

        - ExternalInterface (5)

	Name	RefBaseClassPath
1	ConnectionPoint	AutomationMLInterfaceClassLib/.../PortConnector
2	Output1	AutomationMLInterfaceClassLib/.../SignalInterface
3	Output2	AutomationMLInterfaceClassLib/.../SignalInterface
4	Input1	AutomationMLInterfaceClassLib/.../SignalInterface
5	Input2	AutomationMLInterfaceClassLib/.../SignalInterface

        - RoleRequirements
          - RefBaseRoleClassPath: AutomationMLBaseRoleClassLib/.../Port
      - RoleRequirements
        - RefBaseRoleClassPath: AutomationMLBaseRoleClassLib/.../Resource
    - InternalElement
      - Name: Conveyor2
      - ID: GUID4
      - InternalElement
        - Name: Port
        - ID: GUID5
        - Attribute

Name	Value
Direction	In
xs:string	

        - ExternalInterface (5)

	Name	RefBaseClassPath
1	ConnectionPoint	AutomationMLInterfaceClassLib/.../PortConnector
2	Input1	AutomationMLInterfaceClassLib/.../SignalInterface
3	Input2	AutomationMLInterfaceClassLib/.../SignalInterface
4	Output1	AutomationMLInterfaceClassLib/.../SignalInterface
5	Output2	AutomationMLInterfaceClassLib/.../SignalInterface

        - RoleRequirements
          - RefBaseRoleClassPath: AutomationMLBaseRoleClassLib/.../Port
      - RoleRequirements
        - RefBaseRoleClassPath: AutomationMLBaseRoleClassLib/.../Resource
    - InternalLink (1)

	Name	RefPartnerSideA	RefPartnerSideB
1	L1	GUID3:ConnectionPoint	GUID5:ConnectionPoint

Figure A.11 – XML description of the AML Port concept

```
<InstanceHierarchy Name="PortExample">
  <InternalElement Name="Station" ID="GUID1">
    <InternalElement Name="Conveyor1" ID="GUID2">
      <InternalElement Name="Port" ID="GUID3">
        <Attribute Name="Direction" AttributeDataType="xs:string">
          <Value>Out</Value>
        </Attribute>
        <ExternalInterface Name="ConnectionPoint" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PortConnector"/>
        <ExternalInterface Name="Output1" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface"/>
        <ExternalInterface Name="Output2" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface"/>
        <ExternalInterface Name="Input1" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface"/>
        <ExternalInterface Name="Input2" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Port"/>
      </InternalElement>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
    </InternalElement>
    <InternalElement Name="Conveyor2" ID="GUID4">
      <InternalElement Name="Port" ID="GUID5">
        <Attribute Name="Direction" AttributeDataType="xs:string">
          <Value>In</Value>
        </Attribute>
        <ExternalInterface Name="ConnectionPoint" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PortConnector"/>
        <ExternalInterface Name="Input1" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface"/>
        <ExternalInterface Name="Input2" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface"/>
        <ExternalInterface Name="Output1" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface"/>
        <ExternalInterface Name="Output2" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Port"/>
      </InternalElement>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
    </InternalElement>
    <InternalLink Name="L1" RefPartnerSideA="GUID3:ConnectionPoint" RefPartnerSideB="GUID5:ConnectionPoint"/>
  </InternalElement>
</InstanceHierarchy>
```

Figure A.12 – XML text describing the AML Port concept

A.2.2.3 Modelling a Port as user-defined AML SystemUnitClass

The following example in Figure A.13 depicts an XML description of a user-defined System-UnitClass “myPortClass”.

SystemUnitClass		
= Name	myPortClass	
Attribute (2)		
	= Name	() Value
1	Direction	InOut
2	Category	MaterialFlow
ExternalInterface		
	= Name	ConnectionPoint
	= RefBaseClassPath	AutomationMLInterfaceClassLib/.../PortConnector
SupportedRoleClass		
	= RefRoleClassPath	AutomationMLBaseRoleClassLib/.../Port

```
<SystemUnitClass Name="myPortClass">
  <Attribute Name="Direction">
    <Value>InOut</Value>
  </Attribute>
  <Attribute Name="Category">
    <Value>MaterialFlow</Value>
  </Attribute>
  <ExternalInterface Name="ConnectionPoint" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PortConnector"/>
  <SupportedRoleClass RefRoleClassPath="AutomationMLBaseRoleClassLib/.../Port"/>
</SystemUnitClass>
```

Figure A.13 – Definition of a user-defined AML Port class “myPortClass”

A.2.3 AML Facet concept

A.2.3.1 Concept description

A Facet is an AML object providing a sub-view on attributes or interfaces of the parent AML object. This concept serves for the storage of different configuration settings such as HMI or PLC related data and allows the automation of several control engineering steps. For this, AML defines the AML RoleClass “Facet" (see 6.4.4). Normative provisions are specified in 8.3.

The described subgroup of attributes and interfaces is related to a certain engineering aspect and may store information about corresponding engineering solutions or templates. The syntax or semantics of these attribute names or values is not part of this part of the IEC 62714 and is interpreted by an external engineering tool which has knowledge about the syntax and semantics of the corresponding information. Therefore, these algorithms only need the required Facet information to perform automated engineering tasks. For example, consider that the attributes of an object comprise a name of a PLC code template and the interfaces describe inputs or outputs of or to this template. Thus, a PLC code generation algorithm, that has knowledge about the semantics of these attributes and interfaces, may generate a PLC code out of this information. The same is possible with HMI templates. The mentioned external algorithms or the semantic of corresponding attributes or interfaces are outside of the scope of IEC 62714. In combination with the AML Group concept, an automation of engineering steps may be achieved.

A.2.3.2 Example

Figure A.14 explains the AML Facet concept by means of an example: the object “Conveyor1” comprises the attributes “A” and “B” as well as the interfaces “X” and “Y”. The assigned Facet object “PLCFacet” refers to the attribute “A” and the interface “X”, whereas the assigned Facet object “HMIFacet” refers to the attributes “A” and “B” as well as to the interface “Y”. Hence, both Facets provide a filtered view on certain engineering information which is relevant for different engineering tasks.

Use case: The attribute “A” maybe the name of a vendor specific PLC code template describing the functionality of the object “Conveyor1”. The interface “X” may be the name of an input signal required for this code template. The attribute “B” may be the name of a specific HMI template for the conveyor and the interface “Y” may be a signal that should be presented on the HMI. With this information, a PLC or HMI generator is able to generate solutions automatically. This is exemplarily described in A.2.4.4.

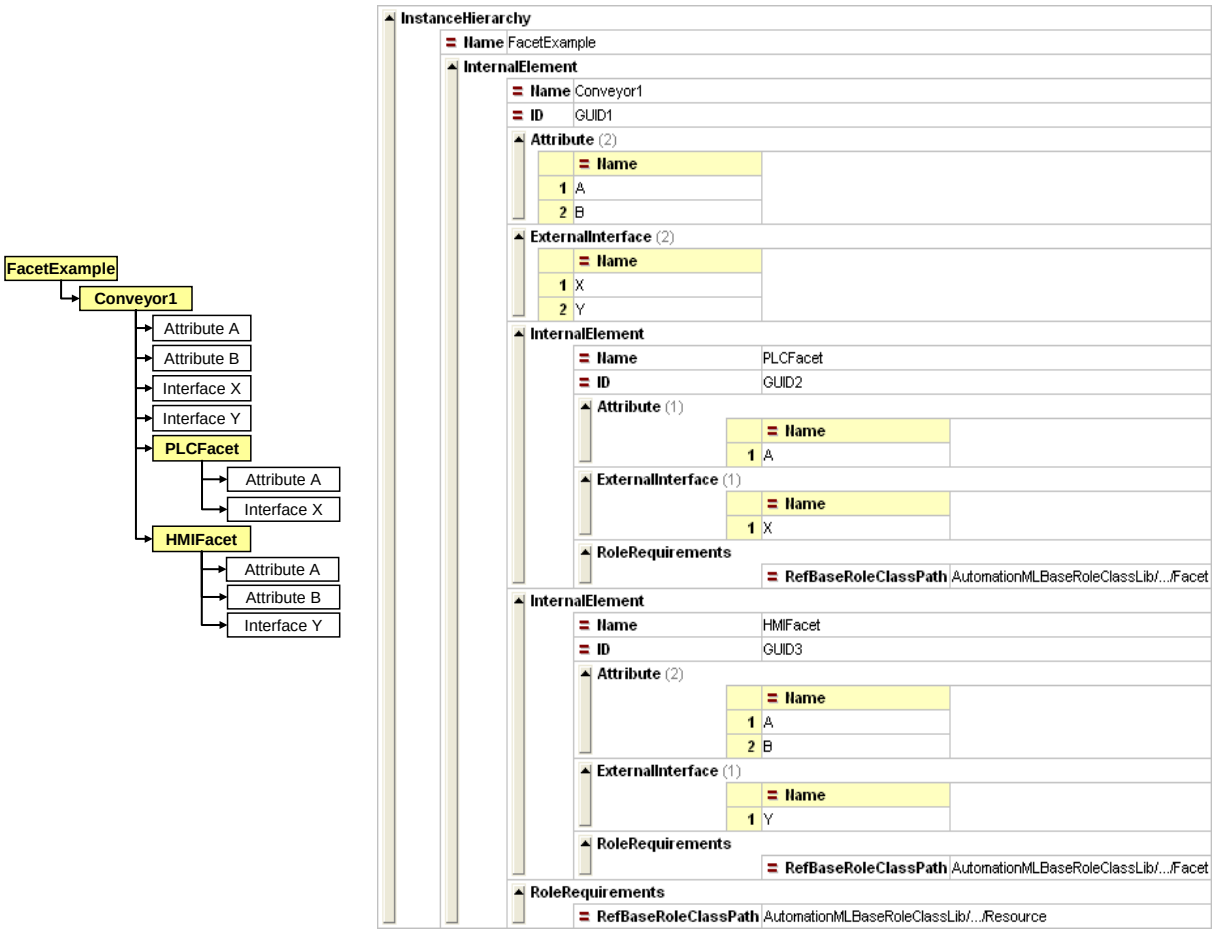


Figure A.14 – AML Facet example

```
<InstanceHierarchy Name="FacetExample">
  <InternalElement Name="Conveyor1" ID="GUID1">
    <Attribute Name="A"/>
    <Attribute Name="B"/>
    <ExternalInterface Name="X"/>
    <ExternalInterface Name="Y"/>
    <InternalElement Name="PLCFacet" ID="GUID2">
      <Attribute Name="A"/>
      <ExternalInterface Name="X"/>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Facet"/>
    </InternalElement>
    <InternalElement Name="HMIFacet" ID="GUID3">
      <Attribute Name="A"/>
      <Attribute Name="B"/>
      <ExternalInterface Name="Y"/>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Facet"/>
    </InternalElement>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
  </InternalElement>
</InstanceHierarchy>
```

Figure A.15 – XML text of the AML Facet example

A.2.4 AML Group concept

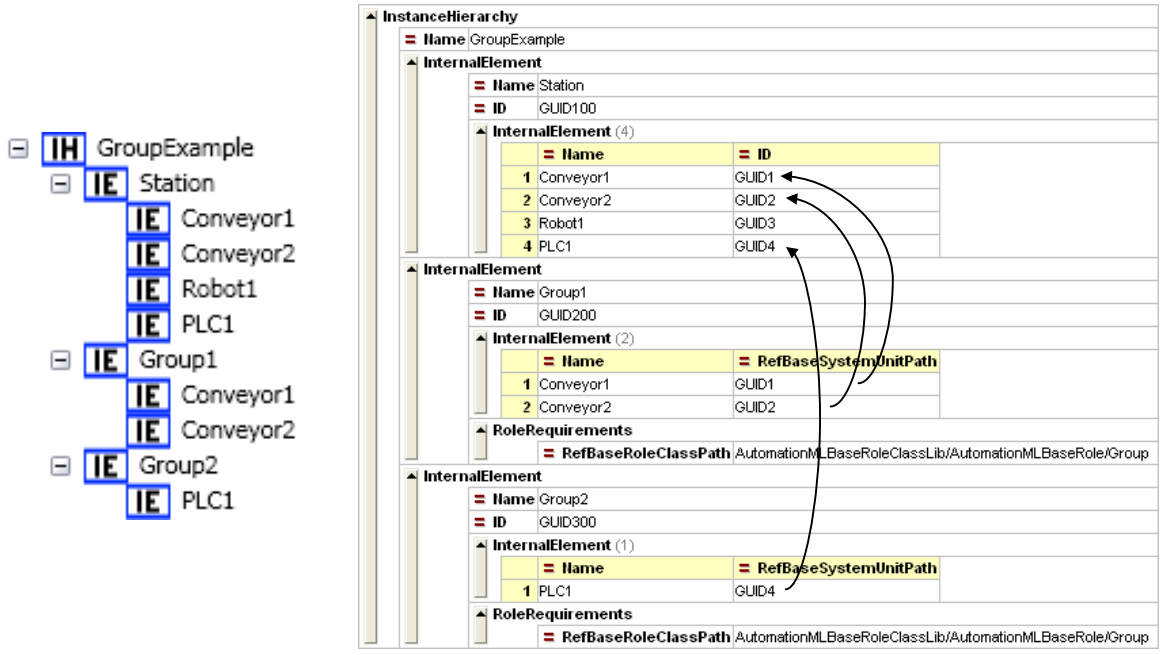
A.2.4.1 Concept description

The AML Group concept allows separating structure information from instance information. Since different engineering tools in a heterogeneous tool landscape may require different views on the same data, it might be useful to store these views separately. This is possible using the AML Group concept and allows structuring identical objects in different hierarchies.

By defining the Group attribute „AssociatedFacet“, a Group can be associated with a type of Facets characterized by a unique name. This allows external engineering algorithms to automatically identify related objects and their corresponding Facets in order to derive engineering information. For this, AML defines the AML RoleClass “Group” (see 6.4.3). Normative provisions are specified in 8.4.

A.2.4.2 Example

Figure A.16a) describes the Group concept by means of a structure “Station” that contains the objects “Conveyor1”, “Conveyor2”, “Robot1” and “PLC1”. Additionally, the objects “Group1” and “Group2” describe the same data in different hierarchies: “Group1” gives a structure view on conveyors only, whereas “Group2” only depicts PLC relevant objects. According to IEC 62424:2008, A.2.14, CAEX provides the storage of such crossed structures. Figure A.16b) gives an AML implementation of this example and Figure A.17 provides the corresponding XML text. The combination between the Facet concept and the Port concept is described in A.2.4.3.



a) Hierarchy b) Grid view

Figure A.16 – AML Group example

```
<InstanceHierarchy Name="GroupExample">
  <InternalElement Name="Station" ID="GUID100">
    <InternalElement Name="Conveyor1" ID="GUID1"/>
    <InternalElement Name="Conveyor2" ID="GUID2"/>
    <InternalElement Name="Robot1" ID="GUID3"/>
    <InternalElement Name="PLC1" ID="GUID4"/>
  </InternalElement>
  <InternalElement Name="Group1" ID="GUID200">
    <InternalElement Name="Conveyor1" RefBaseSystemUnitPath="GUID1"/>
    <InternalElement Name="Conveyor2" RefBaseSystemUnitPath="GUID2"/>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Group"/>
  </InternalElement>
  <InternalElement Name="Group2" ID="GUID300">
    <InternalElement Name="PLC1" RefBaseSystemUnitPath="GUID4"/>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Group"/>
  </InternalElement>
</InstanceHierarchy>
```

Figure A.17 – XML text for the AML Group example

A.2.4.3 Combination of the Group and Facet concept

Figure A.18 presents an example with a combination of the Group and the Facet concept. The shown InstanceHierarchy depicts an AML object “Station” which comprises the AML objects “Conveyor1” and “Conveyor2”. These conveyors each own two attributes and two interfaces.

The AML object “Group” presents nested groups “Group1” and “Group2”. Both refer to the conveyor objects, but have different Facet associations.

Use case: A control code generation algorithm may run through the InstanceHierarchy identifying all groups with an association to a “PLCFacet” and then perform the code generation evaluating the referenced objects.

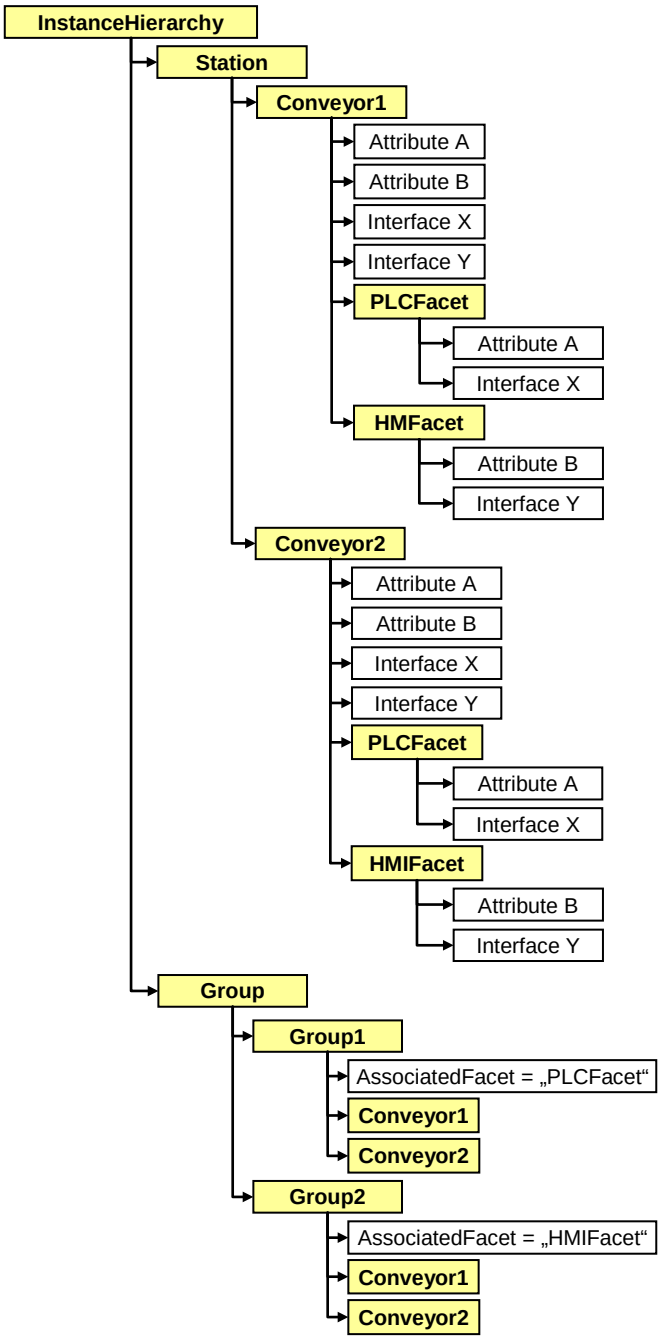


Figure A.18 – Combination of the Facet and Group concept

Figure A.19 shows the corresponding XML text related to the example shown in Figure A.18.

```
<InstanceHierarchy Name="FacetGroupCombination">
  <InternalElement Name="Conveyor1" ID="GUID1">
    <Attribute Name="A"/>
    <Attribute Name="B"/>
    <ExternalInterface Name="X"/>
    <ExternalInterface Name="Y"/>
    <InternalElement Name="PLCFacet" ID="GUID2">
      <Attribute Name="A"/>
      <ExternalInterface Name="X"/>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Facet"/>
    </InternalElement>
    <InternalElement Name="HMIFacet" ID="GUID3">
      <Attribute Name="B"/>
      <ExternalInterface Name="Y"/>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Facet"/>
    </InternalElement>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource"/>
  </InternalElement>
  <InternalElement Name="Conveyor2" ID="GUID4">
    <Attribute Name="A"/>
    <Attribute Name="B"/>
    <ExternalInterface Name="X"/>
    <ExternalInterface Name="Y"/>
    <InternalElement Name="PLCFacet" ID="GUID5">
      <Attribute Name="A"/>
      <ExternalInterface Name="X"/>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Facet"/>
    </InternalElement>
    <InternalElement Name="HMIFacet" ID="GUID6">
      <Attribute Name="B"/>
      <ExternalInterface Name="Y"/>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Facet"/>
    </InternalElement>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource"/>
  </InternalElement>
  <InternalElement Name="Group" ID="GUID7">
    <InternalElement Name="Group1" ID="GUID8">
      <Attribute Name="AccociatedFacet">
        <Value>PLCFacet</Value>
      </Attribute>
      <InternalElement Name="Conveyor1" RefBaseSystemUnitPath="GUID1"/>
      <InternalElement Name="Conveyor2" RefBaseSystemUnitPath="GUID2"/>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Group"/>
    </InternalElement>
    <InternalElement Name="Group2" ID="GUID9">
      <Attribute Name="AccociatedFacet">
        <Value>HMIFacet</Value>
      </Attribute>
      <InternalElement Name="Conveyor1" RefBaseSystemUnitPath="GUID1"/>
      <InternalElement Name="Conveyor2" RefBaseSystemUnitPath="GUID2"/>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Group"/>
    </InternalElement>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Group"/>
  </InternalElement>
</InstanceHierarchy>
```

Figure A.19 – XML text view for the combined Facet-Group example

A.2.4.4 Automatic generation of HMI using the Group and Facet concept

Based on the given example, it is assumed that the conveyor's attribute "B" represents an HMI template visualizing the variable "Y". Figure A.20 illustrates this generic HMI template of a conveyor.

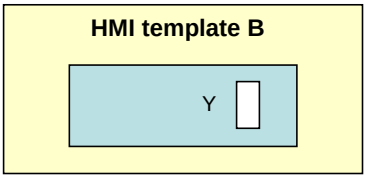


Figure A.20 – Generic HMI template "B" visualizing a process variable "Y" of a conveyor

Based on the concrete conveyor instances, an engineering algorithm is enabled to identify that the AML object "Group2" is associated to the HMI Facet. Here, it identifies that the instances "Conveyor1" and "Conveyor2" are part of the HMI. The algorithm can extract the HMI relevant information of each of both conveyors, can identify the corresponding HMI template and can associate the correct signals to be visualized. Figure A.21 represents the resulting HMI.

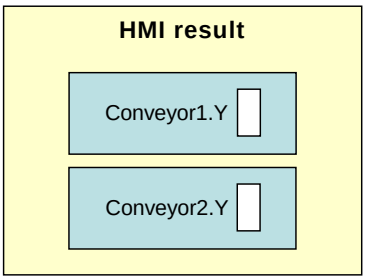


Figure A.21 – Generated HMI result "B" visualizing both conveyors with individual process variables

A.2.5 PropertySet concept

A.2.5.1 Concept description

A PropertySet acts as an attribute taxonomy, a structured dictionary. It is modelled as a role class containing predefined attributes describing properties of a certain scope and is derived from the standard role class "PropertySet" (see 6.4.13). It comprises a list of syntactically and semantically agreed attributes. Property-sets are collected in role class libraries.

AML objects can be associated to one or several property-sets. For each property-set, a separate child object of type CAEX InternalElement should be created and assigned to the corresponding PropertySet class by means of its RoleRequirements definition. Multiple property-sets can be associated by means of multiple child InternalElements of the corresponding AML object.

The CAEX mapping object allows the mapping of the proprietary attributes of the AML object with semantically predefined attributes of the PropertySet. This allows importer software to automatically interpret these attributes and to map them to target tool specific attributes. This simplifies the automatic data exchange across different tools.

A predefined AML library of property-sets can be specified. Normative provisions are described in 8.5.

A.2.5.2 Example

As an example, the layout of an assembly line with a collection of assembly stations (see Figure A.22) is transferred by means of AML. The assembly station is composed of different areas, one for the transport of material, one for the storage of material and one for the assembly, as shown below. The source engineering tool defines these areas with user-defined attributes, assigned to the object, representing the station.

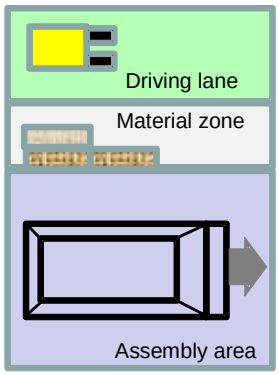


Figure A.22 – PropertySet example

Figure A.23 illustrates the corresponding AML model. The left side of the figure depicts the instance hierarchy for Station1 modelling the three areas. Each of the areas has a user-defined set of attributes and a reference to the PropertySet “Area”. The corresponding XML text is shown in Figure A.24.

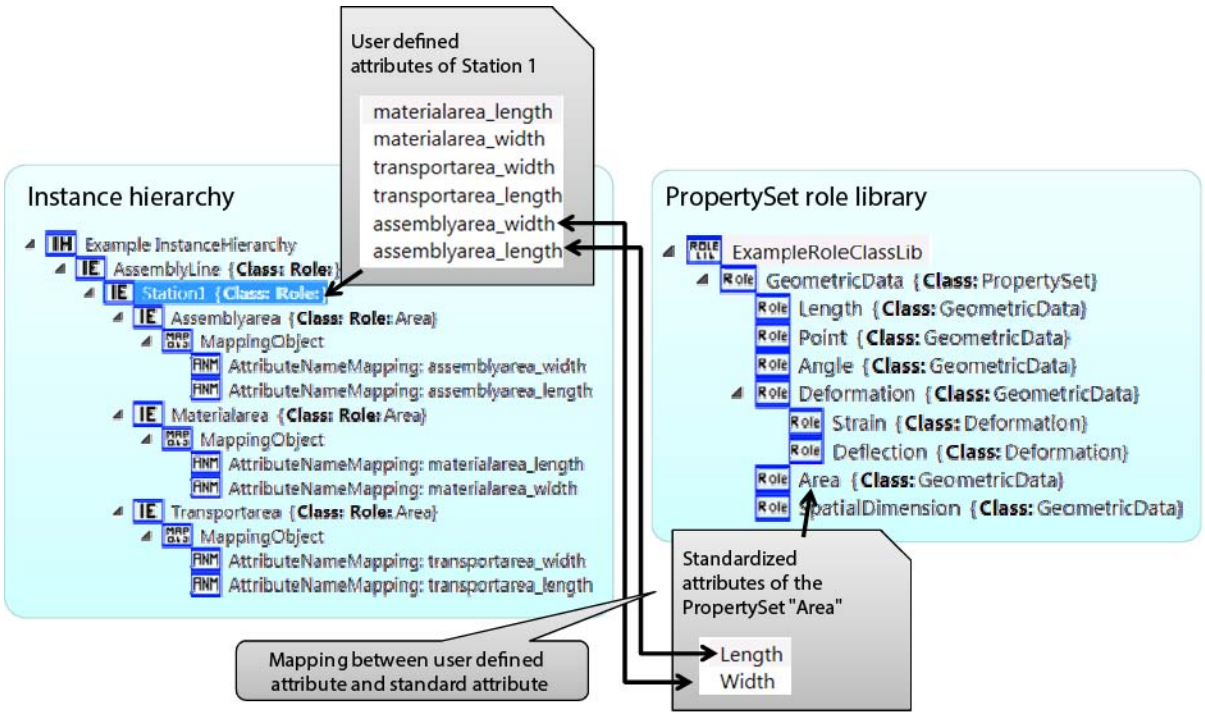


Figure A.23 – PropertySet example

```
<InstanceHierarchy Name="Example InstanceHierarchy">
  <InternalElement Name="AssemblyLine" ID="{8b2510b4-7fc5-4f54-9d09-a3197a062604}">
    <InternalElement Name="Station1" ID="{54500138-c6a1-47c8-80c9-bee35662563c}">
      <Attribute Name="materialarea_length" />
      <Attribute Name="materialarea_width" />
      <Attribute Name="transportarea_width" />
      <Attribute Name="transportarea_length" AttributeDataType="xs:double" />
      <Attribute Name="assemblyarea_width" AttributeDataType="xs:double" />
      <Attribute Name="assemblyarea_length" AttributeDataType="xs:double" />
      <InternalElement Name="Assemblyarea" ID="{e08f53e5-eb0f-45c8-b42f-4714824dd05c}">
        <RoleRequirements RefBaseRoleClassPath="ExampleRoleClassLib/GeometricData/Area" />
        <MappingObject>
          <AttributeNameMapping SystemUnitAttributeName="assemblyarea_width" RoleAttributeName="Width" />
          <AttributeNameMapping SystemUnitAttributeName="assemblyarea_length" RoleAttributeName="Length" />
        </MappingObject>
      </InternalElement>
      <InternalElement Name="Materialarea" ID="{668360af-d108-4b60-be53-ddb262bc470e}">
        <RoleRequirements RefBaseRoleClassPath="ExampleRoleClassLib/GeometricData/Area" />
        <MappingObject>
          <AttributeNameMapping SystemUnitAttributeName="materialarea_length" RoleAttributeName="Length" />
          <AttributeNameMapping SystemUnitAttributeName="materialarea_width" RoleAttributeName="Width" />
        </MappingObject>
      </InternalElement>
      <InternalElement Name="Transportarea" ID="{19418967-cd7c-46ea-adea-81aa52f6b685}">
        <RoleRequirements RefBaseRoleClassPath="ExampleRoleClassLib/GeometricData/Area" />
        <MappingObject>
          <AttributeNameMapping SystemUnitAttributeName="transportarea_width" RoleAttributeName="Width" />
          <AttributeNameMapping SystemUnitAttributeName="transportarea_length" RoleAttributeName="Length" />
        </MappingObject>
      </InternalElement>
    </InternalElement>
  </InternalElement>
</InstanceHierarchy>
```

Figure A.24 – XML text for the instance hierarchy

The right side of Figure A.23 illustrates a sample AML PropertySet library. This AML RoleClass library contains the RoleClass “GeometricData”, which is derived from the Base-RoleClass “PropertySet”. The RoleClass “GeometricData” itself has additional derivations, which define some basic geometric properties. The RoleClass named “Area” defines the attributes “Length” and “Width” as attributes of Type “double” and Unit “m [metre]”. The XML text in Figure A.25 shows the definitions of the attributes of these PropertySets.

```
- <RoleClass Name="GeometricData"
    RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/PropertySet
  - <RoleClass Name="Length"
      RefBaseClassPath="ExampleRoleClassLib/GeometricData">
      <Attribute Name="lengthValue" AttributeDataType="xs:double"
        Unit="m" />
    </RoleClass>
  - <RoleClass Name="Point"
      RefBaseClassPath="ExampleRoleClassLib/GeometricData">
      <Attribute Name="xAxis" AttributeDataType="xs:double" />
      <Attribute Name="yAxis" AttributeDataType="xs:double" />
      <Attribute Name="zAxis" AttributeDataType="xs:double" />
    </RoleClass>
  - <RoleClass Name="Angle"
      RefBaseClassPath="ExampleRoleClassLib/GeometricData">
      <Attribute Name="angleValue" AttributeDataType="xs:double"
        Unit="rad" />
    </RoleClass>
  - <RoleClass Name="Deformation"
      RefBaseClassPath="ExampleRoleClassLib/GeometricData">
      <Description>Deformation of the material is the change in geometry
        when stress is applied (in the form of force loading, gravitational
        field, acceleration, thermal expansion, etc.). Deformation is
        expressed by the displacement field of the material</Description>
      <Attribute Name="deformationValue" AttributeDataType="xs:double" />
    - <RoleClass Name="Strain"
        RefBaseClassPath="ExampleRoleClassLib/GeometricData/Deformation">
        <Description>Strain is the deformation per unit length</Description>
      </RoleClass>
    - <RoleClass Name="Deflection"
        RefBaseClassPath="ExampleRoleClassLib/GeometricData/Deformation">
        <Description>Deflection is a term to describe the magnitude to
          which a structural element bends under a load</Description>
      </RoleClass>
    </RoleClass>
  - <RoleClass Name="Area"
      RefBaseClassPath="ExampleRoleClassLib/GeometricData">
      <Attribute Name="Length" AttributeDataType="xs:double" Unit="m" />
      <Attribute Name="Width" AttributeDataType="xs:double" Unit="m" />
    </RoleClass>
  - <RoleClass Name="SpatialDimension"
      RefBaseClassPath="ExampleRoleClassLib/GeometricData">
      <Attribute Name="XAxis" AttributeDataType="xs:double" Unit="m" />
      <Attribute Name="YAxis" AttributeDataType="xs:double" Unit="m" />
      <Attribute Name="ZAxis" AttributeDataType="xs:double" Unit="m" />
    </RoleClass>
  </RoleClass>
```

Figure A.25 – PropertySet example AML library as XML code

With the use of PropertySets the exporting tool can store user-defined objects with user-defined attributes and explain the semantic of the user-defined attributes by means of mappings. This supports the automatic interpretation of those attributes. As a result, a target engineering tool could interpret the data and can perform unit-conversions to the required units of the PropertySet attributes.

A.2.6 Process-Product-Resource concept

A.2.6.1 Concept description

In the course of structuring complex plant engineering data, the trisection of the data into resources, processes and products has delivered a proven performance in practice. This concept is applied in different fields, e.g. for digital factory tools or with IEC 62264 at the manufacturing execution system (MES) level.

- In a resource centric view, resources form the central component within the model: they execute processes and handle products. In AML, a resource is an entity involved in

production including plants, robots, machines, their state, equipment, possible messages and so on. According to this, resources can be hardware components of a production system, as well as software systems, e.g. SCADA systems. Within AML, resources are typically modelled in a plant hierarchy forming the plant topology.

- In a product centric view, the produced product is the focus of consideration. It determines which processes should be applied to the materials or intermediate products and which equipment should be used therefore. This is valid in the field of continuous, discrete or batch control. A product in AML depicts a produced good. It can be built up hierarchically. It is essential that products do not have to be final products. Test results belong to products as well as product data and the corresponding documentation.
- In a process centric view, processes form the central items of the model. A process in AML represents a production process including sub-processes. Process parameters, the process chain and the process planning form part of the processes. In technical terms, processes modify products. This corresponds to the usage in AML as final products are produced out of different sub-products, or chemical treatments change substances. However, processes have relations to resources and vice versa.

In every case, representations of the resource, product and process are linked to each other (see Figure A.26). One reasonable assignment to the process “transport” is the resource “conveyor“. A “press” could create “scrap cubes“. And a “welding” process can “weld” two “metals” together.

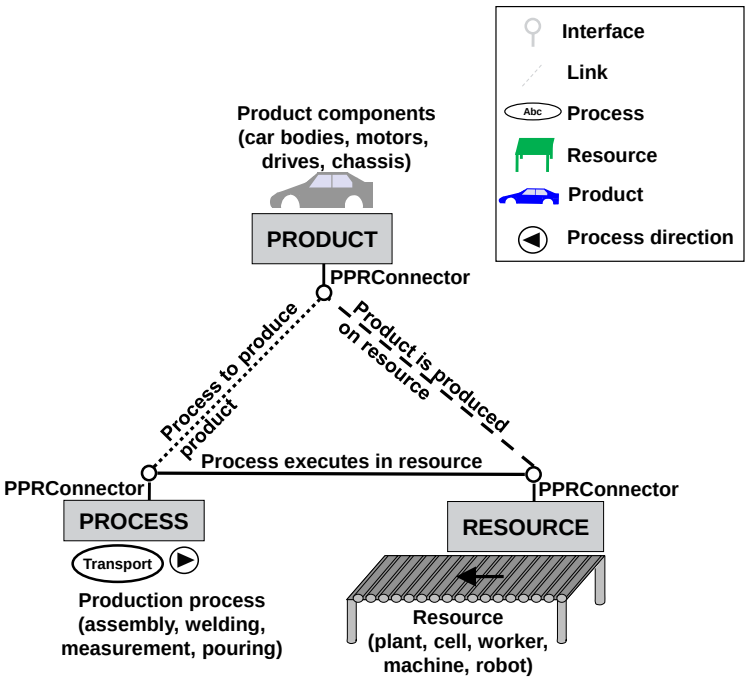


Figure A.26 – Base elements of the Product-Process-Resource concept

To create a link between these elements, they require an interface. For this, AML defines the standard interface class PPRConnector (see Figure A.27). Normative provisions regarding the PPRConnector are specified in 6.3.5. By this interface, links can be established between the elements by means of standard CAEX InternalLinks (see 5.6.6). Thus, resources can be linked to products which they can manipulate.

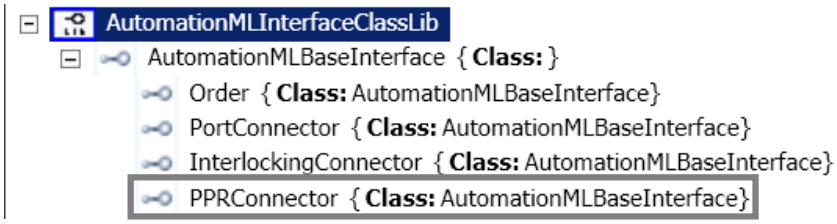


Figure A.27 – PPRConnector interface

A.2.6.2 Example

The following example (see Figure A.28) illustrates the application of this concept with AML. It consists of two conveyers (C1 and C2), a turntable (TT1) and a robot (RB1). These are the resources of the plant. The robot assembles wheels to the cars. The wheels as well as the cars are products. Production processes within the example are transport, turn and assemble.

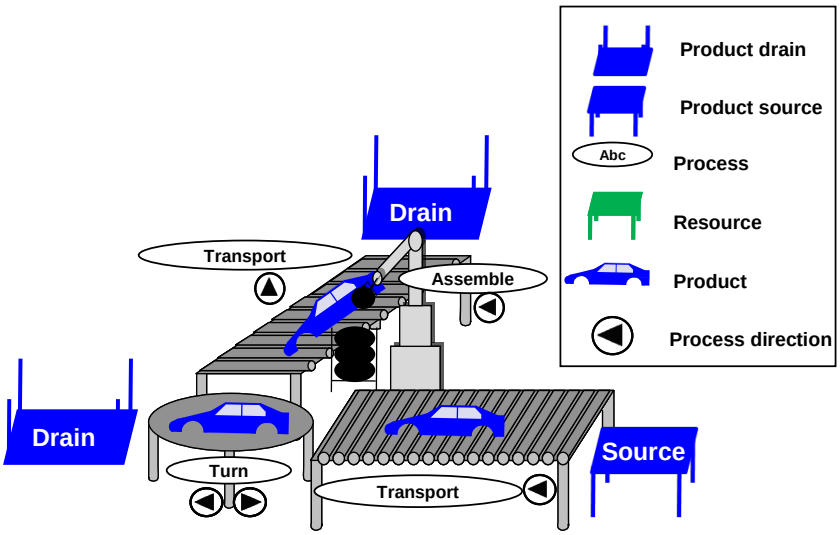


Figure A.28 – Example for the Product-Process-Resource concept

In AML, the Process-Product–Resource-concept is modelled by means of the CAEX role concept (see IEC 62424:2008, A.2.9) and relations between the elements (see 5.6.6). The sets of elements with assigned roles “Resource”, “Product” or “Process” are pairwise mutually exclusive. This means that a resource cannot be a product or a process at the same time. The corresponding role classes are part of the AutomationMLBaseRoleClassLib (see Figure A.29 and 6.4).

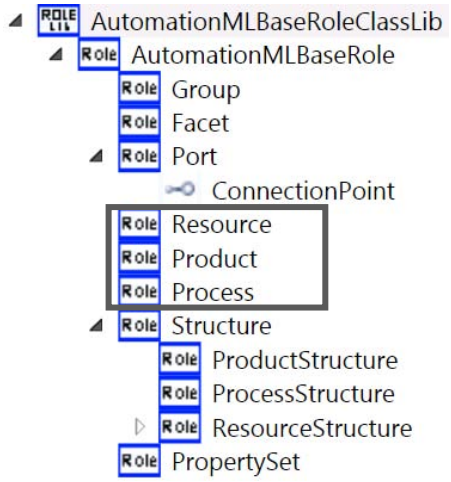


Figure A.29 – AML roles required for the Process-Product-Resource concept

In the example (see Figure A.28), the role “Resource” is assigned to the conveyors, the robot and the turntable. Cars and wheels are assigned to the role “Product” and the role “Process” is assigned to transport, turn and assemble process elements. All elements are stored in the corresponding sub-tree which can be seen in Figure A.30. The order of processes, products or resources can be explicitly expressed by links between corresponding interfaces of type “Order” (this is not depicted in this example due to readability reasons).

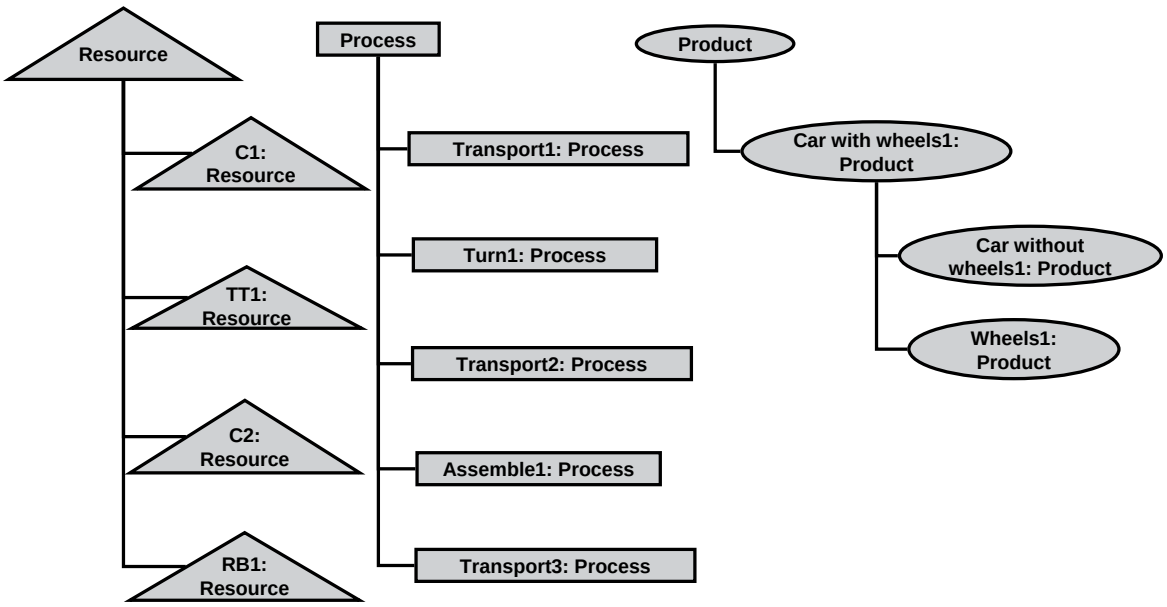


Figure A.30 – Elements of the example

Each element in the example has a PPRConnector interface. The complete links of the example are represented in Figure A.31. The solid lines represent links from resources to processes, the dotted lines links from processes to products and the dashed lines are links between resources and products. This reveals the complexity. Thus, redundant connections can be omitted.

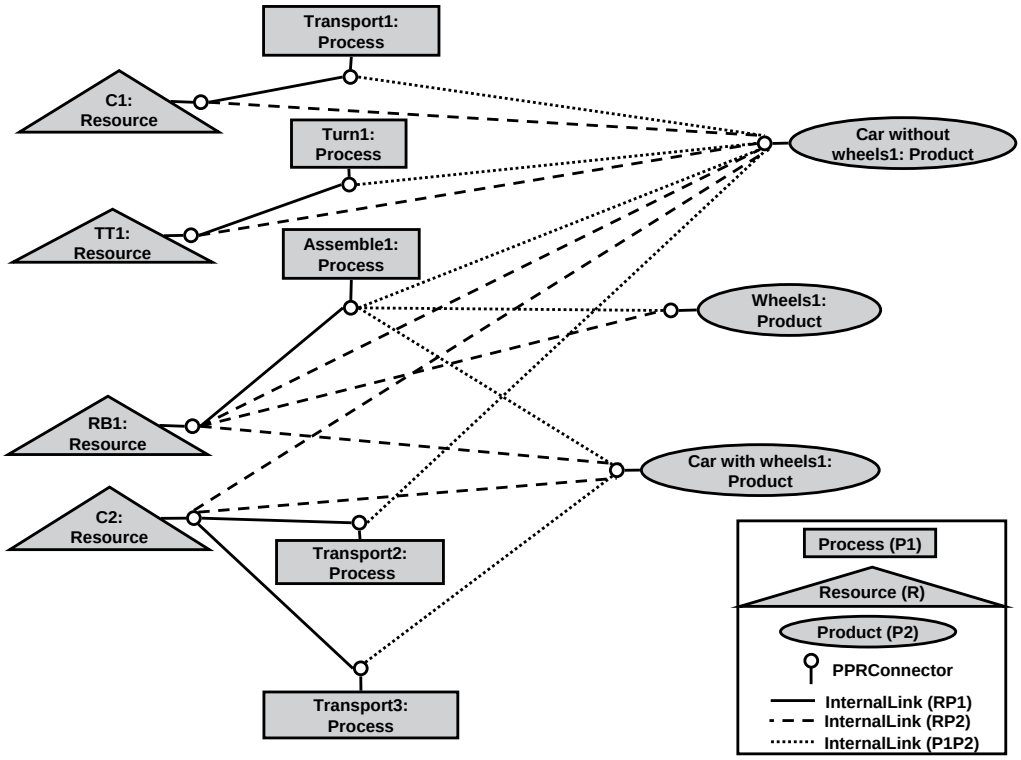


Figure A.31 – Links within the example

Figure A.32 shows the resource centric view on the considered example. Therefore, only twelve instead of nineteen links are necessary. The conveyor “C1” is connected with the product “Car without wheels1”as are the turn table “TT1” and the conveyor “C2”. As the robot assembles the wheels to the cars on the conveyor “C2”, the robot “RB1” is linked to the “Car without wheels1“, the “Car with wheels1” and the “Wheels1”. Additionally, the conveyor “C2” has a link to the “Car with wheels1”. The process “Transport1” is assigned to “C1”, and “Transport2” and “Transport3” are connected to the conveyor “C2”. “Assemble1” is related to the robot “RB1” and “Turn1” to the turn table “TT1”. The links from the products to the processes (dotted lines in Figure A.31) can be derived from the existing links. The model can be arbitrarily rotated and arranged to centre the elements of type product or process.

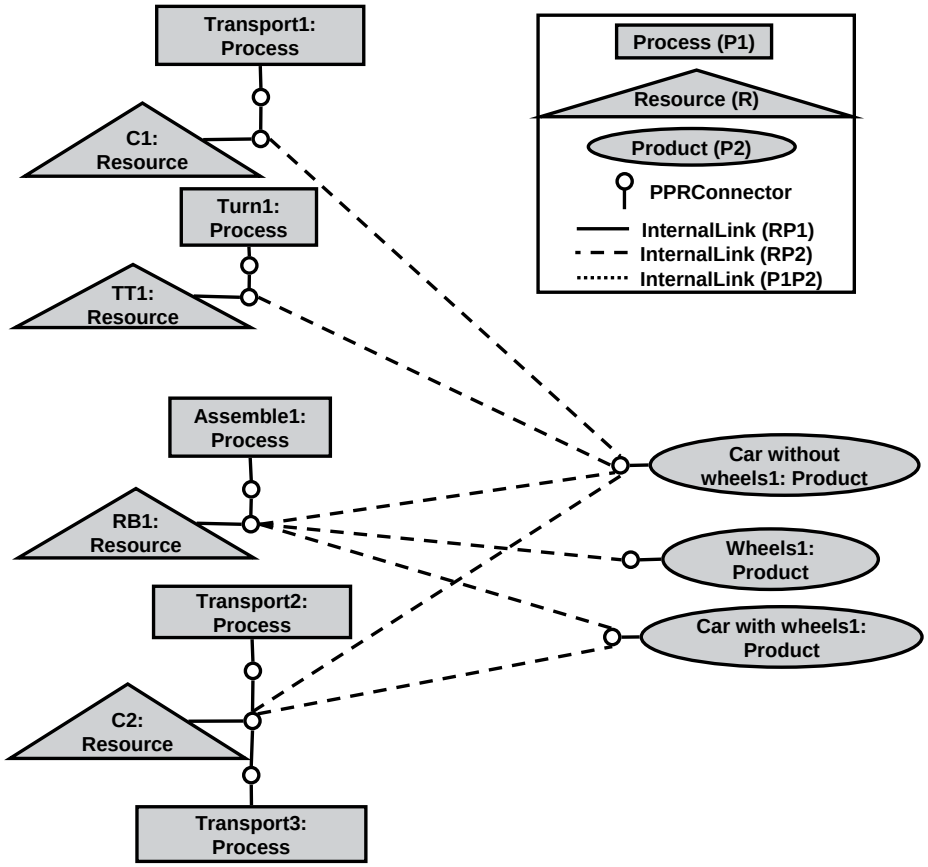


Figure A.32 – Links of the resource centric view on the example

Figure A.33 illustrates the AML object tree including a highlighted link between the conveyor “C1” and the process “Transport1“.

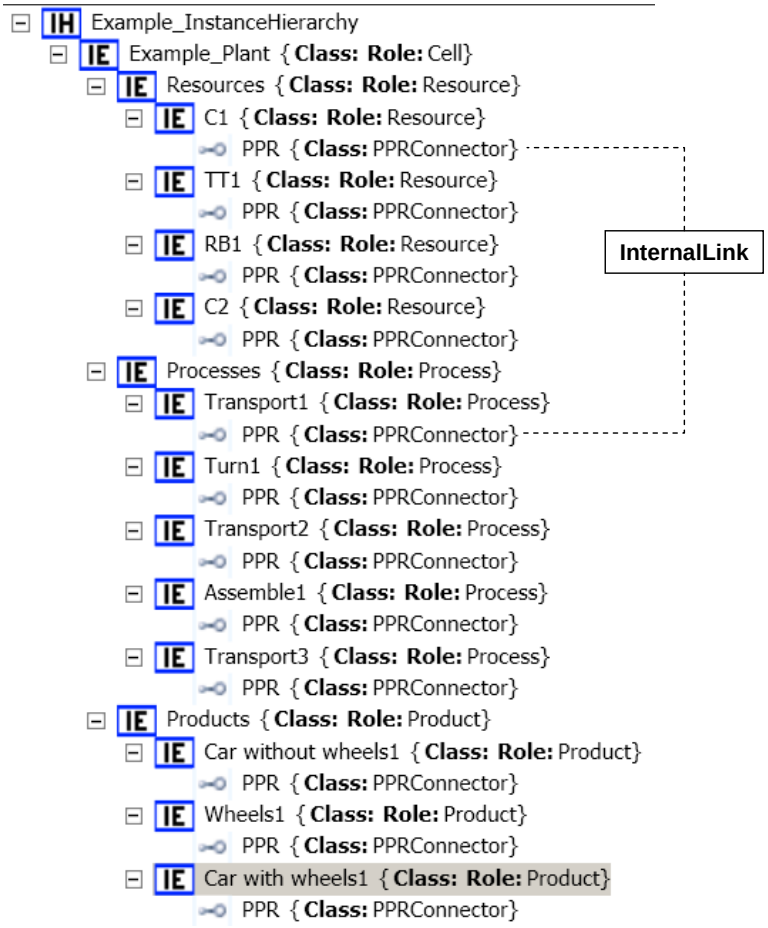


Figure A.33 – InstanceHierarchy of the example in AML

The corresponding XML model is depicted in Figure A.34. On the first level of the example, there are the three basic elements: “Resources”, “Processes” and “Products” modelled as CAEX InternalElement.

InternalElement

= Name	Example_Plant
= ID	GUID1

InternalElement (3)

	= Name	= ID	{} InternalElement
1	Resources	GUID2	InternalElement (4)
2	Processes	GUID7	InternalElement (5)
3	Products	GUID13	InternalElement (3)

InternalElement (4)

	= Name	= ID
1	C1	GUID3
2	TT1	GUID4
3	RB1	GUID5
4	C2	GUID6

InternalElement (5)

	= Name	= ID
1	Transport1	GUID8
2	Turn1	GUID9
3	Transport2	GUID10
4	Assemble1	GUID11
5	Transport3	GUID12

InternalElement (3)

	= Name	= ID
1	Car without wheels1	GUID14
2	Wheels1	GUID15
3	Car with wheels1	GUID16

Figure A.34 – InternalElements of the example

Beneath the object “Resources”, there are the four components of the example: the conveyors, the turntable and the robot. They are of type InternalElement as well. They possess an ExternalInterface PPRConnector and assign the role class “Resource”. The processes and products have an interface and a role assignment as well. To link the elements within the example, the InternalLinks are usually placed on the same level as the most top basic element. The links in XML are depicted as in Figure A.35.

InternalLink (12)

	= Name	= RefPartnerSideA	= RefPartnerSideB
1	C1_T1	GUID3:PPR	GUID8:PPR
2	TT1_Tu1	GUID4:PPR	GUID9:PPR
3	RB1_A1	GUID5:PPR	GUID11:PPR
4	C2_T2	GUID6:PPR	GUID10:PPR
5	C2_T3	GUID6:PPR	GUID12:PPR
6	C1_CwW1	GUID3:PPR	GUID14:PPR
7	TT1_CwW1	GUID4:PPR	GUID14:PPR
8	RB1_CwW1	GUID5:PPR	GUID14:PPR
9	RB1_W1	GUID5:PPR	GUID15:PPR
10	RB1_CW1	GUID5:PPR	GUID16:PPR
11	C2_CwW1	GUID6:PPR	GUID14:PPR
12	C2_CW1	GUID6:PPR	GUID16:PPR

Figure A.35 – InternalLinks of the example

A complete overview of the example can be seen in Figure A.36.

```
<InstanceHierarchy Name="Example_InstanceHierarchy">
  <InternalElement Name="Example_Plant" ID="GUID1">
    <InternalElement Name="Resources" ID="GUID2">
      <InternalElement Name="C1" ID="GUID3">
        <ExternalInterface Name="PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
      </InternalElement>
      <InternalElement Name="TT1" ID="GUID4">
        <ExternalInterface Name="PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
      </InternalElement>
      <InternalElement Name="RB1" ID="GUID5">
        <ExternalInterface Name="PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
      </InternalElement>
      <InternalElement Name="C2" ID="GUID6">
        <ExternalInterface Name="PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
      </InternalElement>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
    </InternalElement>
    <InternalElement Name="Processes" ID="GUID7">
      <InternalElement Name="Transport1" ID="GUID8">
        <ExternalInterface Name="PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Process"/>
      </InternalElement>
      <InternalElement Name="Turn1" ID="GUID9">
        <ExternalInterface Name="PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Process"/>
      </InternalElement>
      <InternalElement Name="Transport2" ID="GUID10">
        <ExternalInterface Name="PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Process"/>
      </InternalElement>
      <InternalElement Name="Assemble1" ID="GUID11">
        <ExternalInterface Name="PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Process"/>
      </InternalElement>
      <InternalElement Name="Transport3" ID="GUID12">
        <ExternalInterface Name="PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Process"/>
      </InternalElement>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Process"/>
    </InternalElement>
    <InternalElement Name="Products" ID="GUID13">
      <InternalElement Name="Car without wheels1" ID="GUID14">
        <ExternalInterface Name="PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Product"/>
      </InternalElement>
      <InternalElement Name="Wheels1" ID="GUID15">
        <ExternalInterface Name="PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Product"/>
      </InternalElement>
      <InternalElement Name="Car with wheels1" ID="GUID16">
        <ExternalInterface Name="PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Product"/>
      </InternalElement>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Product"/>
    </InternalElement>
    <InternalLink Name="C1_T1" RefPartnerSideA="GUID3:PPR" RefPartnerSideB="GUID8:PPR"/>
    <InternalLink Name="TT1_Tu1" RefPartnerSideA="GUID4:PPR" RefPartnerSideB="GUID9:PPR"/>
    <InternalLink Name="RB1_A1" RefPartnerSideA="GUID5:PPR" RefPartnerSideB="GUID11:PPR"/>
    <InternalLink Name="C2_T2" RefPartnerSideA="GUID6:PPR" RefPartnerSideB="GUID10:PPR"/>
    <InternalLink Name="C2_T3" RefPartnerSideA="GUID6:PPR" RefPartnerSideB="GUID12:PPR"/>
    <InternalLink Name="C1_CwW1" RefPartnerSideA="GUID3:PPR" RefPartnerSideB="GUID14:PPR"/>
    <InternalLink Name="TT1_CwW1" RefPartnerSideA="GUID4:PPR" RefPartnerSideB="GUID14:PPR"/>
    <InternalLink Name="RB1_CwW1" RefPartnerSideA="GUID5:PPR" RefPartnerSideB="GUID14:PPR"/>
    <InternalLink Name="RB1_W1" RefPartnerSideA="GUID5:PPR" RefPartnerSideB="GUID15:PPR"/>
    <InternalLink Name="RB1_CW1" RefPartnerSideA="GUID5:PPR" RefPartnerSideB="GUID16:PPR"/>
    <InternalLink Name="C2_CwW1" RefPartnerSideA="GUID6:PPR" RefPartnerSideB="GUID14:PPR"/>
    <InternalLink Name="C2_CW1" RefPartnerSideA="GUID6:PPR" RefPartnerSideB="GUID16:PPR"/>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Cell"/>
  </InternalElement>
</InstanceHierarchy>
```

Figure A.36 – InstanceHierarchy of the example in XML

A.2.7 Support of multiple roles

In addition to IEC 62424:2008, A.2.9, AML defines how to specify multiple role support for an object instance. Multiple roles are of interest, if an object can have multiple functionalities. An

example is a multi-functional device that is a scanner, a printer or a fax device at the same time. Provisions regarding multiple roles are specified in 8.5.

Figure A.37 gives an example where the object “MultiDevice01” has three attributes “FaxBoudRate”, “PrintSpeed” and “FaxSpeed” and two interfaces “PowerSupply” and “USB”. The object “MultiDevice01” supports three roles “Printer”, “Fax” and “Scanner”. The role referenced with the tag “RefBaseRoleClassPath” of the corresponding RoleRequirements element optionally represents the main role. Figure A.38 presents the corresponding XML code.

Attributes and interfaces belonging to the object “MultiDevice01” should be mapped to the attributes and interfaces of all three associated roles. This is done by means of the CAEX MappingObject according to IEC 62424:2008, A.2.10, which provides information about which role-attribute/interface is associated to which instance attribute/interface. In order to distinguish the attributes of the multiple roles (which may have the same name), the role name should be included into the mapping definition – except the main role specified by “RefBaseRoleClassPath”. Figure A.37 presents a corresponding example of how to specify required attributes and interfaces and how to map them against the instance attributes and interfaces.

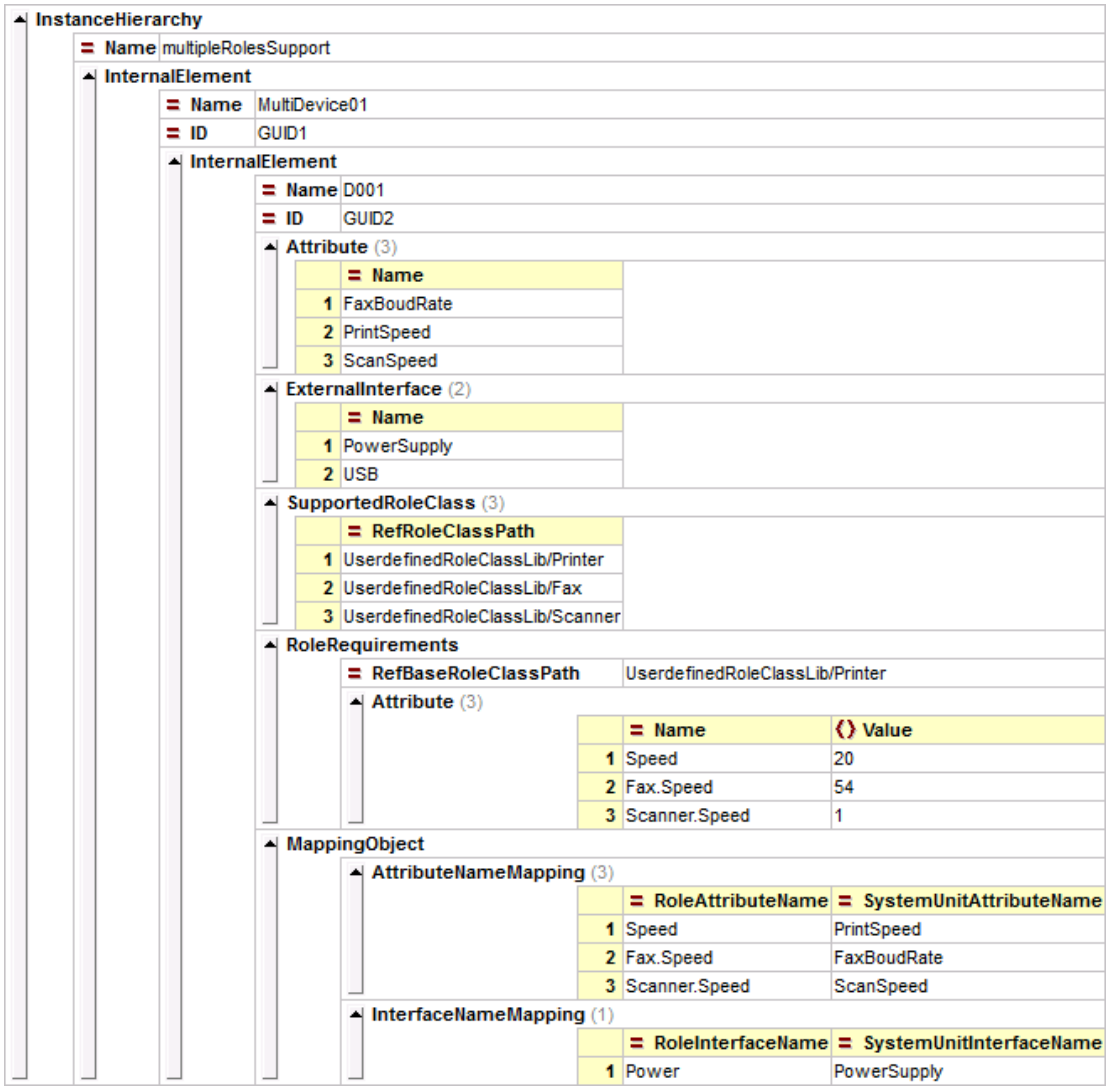


Figure A.37 – Example of a user-defined instance supporting multiple roles

Figure A.38 shows the AML representation of this structure.

```
<InstanceHierarchy Name="multipleRolesSupport">
  <InternalElement Name="MultiDevice01" ID="GUID1">
    <InternalElement Name="D001" ID="GUID2">
      <Attribute Name="FaxBoudRate"/>
      <Attribute Name="PrintSpeed"/>
      <Attribute Name="ScanSpeed"/>
      <ExternalInterface Name="PowerSupply"/>
      <ExternalInterface Name="USB"/>
      <SupportedRoleClass RefRoleClassPath="UserdefinedRoleClassLib/Printer"/>
      <SupportedRoleClass RefRoleClassPath="UserdefinedRoleClassLib/Fax"/>
      <SupportedRoleClass RefRoleClassPath="UserdefinedRoleClassLib/Scanner"/>
      <RoleRequirements RefBaseRoleClassPath="UserdefinedRoleClassLib/Printer">
        <Attribute Name="Speed">
          <Value>20</Value>
        </Attribute>
        <Attribute Name="Fax.Speed">
          <Value>54</Value>
        </Attribute>
        <Attribute Name="Scanner.Speed">
          <Value>1</Value>
        </Attribute>
      </RoleRequirements>
      <MappingObject>
        <AttributeNameMapping RoleAttributeName="Speed" SystemUnitAttributeName="PrintSpeed"/>
        <AttributeNameMapping RoleAttributeName="Fax.Speed" SystemUnitAttributeName="FaxBoudRate"/>
        <AttributeNameMapping RoleAttributeName="Scanner.Speed" SystemUnitAttributeName="ScanSpeed"/>
        <InterfaceNameMapping RoleInterfaceName="Power" SystemUnitInterfaceName="PowerSupply"/>
      </MappingObject>
    </InternalElement>
  </InternalElement>
```

Figure A.38 – XML text of the AML representation of multiple role support

Figure A.39 and Figure A.40 show the corresponding AML role class library as well as its XML representation.

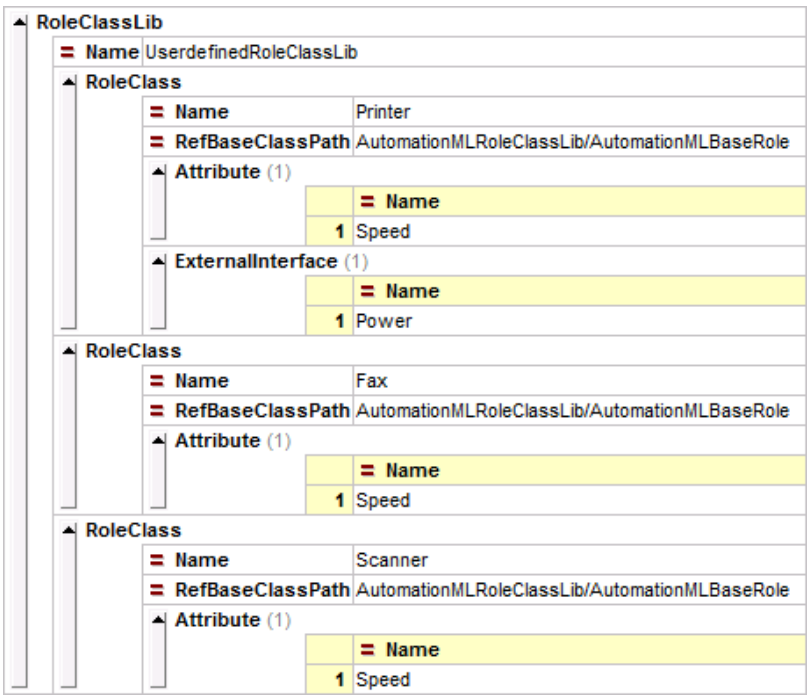


Figure A.39 – AML Role class library corresponding to the multiple role definition example

```
<RoleClassLib Name="UserdefinedRoleClassLib">
  <RoleClass Name="Printer" RefBaseClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole">
    <Attribute Name="Speed"/>
    <ExternalInterface Name="Power"/>
  </RoleClass>
  <RoleClass Name="Fax" RefBaseClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole">
    <Attribute Name="Speed"/>
  </RoleClass>
  <RoleClass Name="Scanner" RefBaseClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole">
    <Attribute Name="Speed"/>
  </RoleClass>
</RoleClassLib>
```

Figure A.40 – XML text of the AML role class library

Annex B
(informative)

XML Representation of AML Libraries

B.1 AutomationMLBaseRoleClassLib

```
<?xml version="1.0" encoding="utf-8"?>
<CAEXFile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd"
   FileName="AutomationMLBaseRoleClassLib.aml" SchemaVersion="2.15">
 <AdditionalInformation AutomationMLVersion="2.0" />
 <AdditionalInformation>
  <WriterHeader>
   <WriterName>IEC SC65E WG 9</WriterName>
   <WriterID>IEC SC65E WG 9</WriterID>
   <WriterVendor>IEC</WriterVendor>
   <WriterVendorURL>www.iec.ch</WriterVendorURL>
   <WriterVersion>1.0</WriterVersion>
   <WriterRelease>1.0.0</WriterRelease>
   <LastWritingDateTime>2013-03-01</LastWritingDateTime>
   <WriterProjectTitle>Automation Markup Language Standard
     Libraries</WriterProjectTitle>
   <WriterProjectID>Automation Markup Language Standard
     Libraries</WriterProjectID>
  </WriterHeader>
 </AdditionalInformation>
 <ExternalReference Path="../InterfaceClass
   Libraries/AutomationMLInterfaceClassLib.aml"
   Alias="AutomationMLInterfaceClassLib" />
 <RoleClassLib Name="AutomationMLBaseRoleClassLib">
  <Description>Automation Markup Language base role class
    library</Description>
  <Version>2.2.0</Version>
  <RoleClass Name="AutomationMLBaseRole">
   <RoleClass Name="Group" RefBaseClassPath="AutomationMLBaseRole">
    <Attribute Name="AssociatedFacet" AttributeDataType="xs:string" />
   </RoleClass>
   <RoleClass Name="Facet" RefBaseClassPath="AutomationMLBaseRole" />
   <RoleClass Name="Port" RefBaseClassPath="AutomationMLBaseRole">
    <Attribute Name="Direction" AttributeDataType="xs:string" />
    <Attribute Name="Cardinality">
     <Attribute Name="MinOccur" AttributeDataType="xs:unsignedInt" />
     <Attribute Name="MaxOccur" AttributeDataType="xs:unsignedInt" />
    </Attribute>
    <Attribute Name="Category" AttributeDataType="xs:string" />
    <ExternalInterface Name="ConnectionPoint" ID="9942bd9c-c19d-44e4-a197-
      11b9edf264e7"
      RefBaseClassPath="AutomationMLInterfaceClassLib@AutomationMLInterfaceC
      lassLib/AutomationMLBaseInterface/PortConnector" />
   </RoleClass>
   <RoleClass Name="Resource" RefBaseClassPath="AutomationMLBaseRole" />
   <RoleClass Name="Product" RefBaseClassPath="AutomationMLBaseRole" />
   <RoleClass Name="Process" RefBaseClassPath="AutomationMLBaseRole" />
   <RoleClass Name="Structure" RefBaseClassPath="AutomationMLBaseRole">
    <RoleClass Name="ProductStructure" RefBaseClassPath="Structure" />
    <RoleClass Name="ProcessStructure" RefBaseClassPath="Structure" />
    <RoleClass Name="ResourceStructure" RefBaseClassPath="Structure" />
   </RoleClass>
   <RoleClass Name="PropertySet" RefBaseClassPath="AutomationMLBaseRole" />
  </RoleClass>
 </RoleClassLib>
</CAEXFile>
```

B.2 AutomationMLInterfaceClassLib

```
<?xml version="1.0" encoding="utf-8"?>
<CAEXFile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd"
FileName="AutomationMLInterfaceClassLib.aml" SchemaVersion="2.15">
 <AdditionalInformation AutomationMLVersion="2.0" />
 <AdditionalInformation>
  <WriterHeader>
   <WriterName>IEC SC65E WG 9</WriterName>
   <WriterID>IEC SC65E WG 9</WriterID>
   <WriterVendor>IEC</WriterVendor>
   <WriterVendorURL>www.iec.ch</WriterVendorURL>
   <WriterVersion>1.0</WriterVersion>
   <WriterRelease>1.0.0</WriterRelease>
   <LastWritingDateTime>2013-03-01</LastWritingDateTime>
   <WriterProjectTitle>Automation Markup Language Standard
    Libraries</WriterProjectTitle>
   <WriterProjectID>Automation Markup Language Standard
    Libraries</WriterProjectID>
  </WriterHeader>
 </AdditionalInformation>
 <InterfaceClassLib Name="AutomationMLInterfaceClassLib">
  <Description>Standard Automation Markup Language Interface Class
   Library</Description>
  <Version>2.2.0</Version>
  <InterfaceClass Name="AutomationMLBaseInterface">
   <InterfaceClass Name="Order" RefBaseClassPath="AutomationMLBaseInterface">
    <Attribute Name="Direction" AttributeDataType="xs:string" />
   </InterfaceClass>
   <InterfaceClass Name="PortConnector"
    RefBaseClassPath="AutomationMLBaseInterface" />
   <InterfaceClass Name="InterlockingConnector"
    RefBaseClassPath="AutomationMLBaseInterface" />
   <InterfaceClass Name="PPRConnector"
    RefBaseClassPath="AutomationMLBaseInterface" />
   <InterfaceClass Name="ExternalDataConnector"
    RefBaseClassPath="AutomationMLBaseInterface">
    <Attribute Name="refURI" AttributeDataType="xs:anyURI" />
    <InterfaceClass Name="COLLADAInterface"
    RefBaseClassPath="ExternalDataConnector" />
    <InterfaceClass Name="PLCopenXMLInterface"
    RefBaseClassPath="ExternalDataConnector" />
   </InterfaceClass>
   <InterfaceClass Name="Communication"
    RefBaseClassPath="AutomationMLBaseInterface">
    <InterfaceClass Name="SignalInterface" RefBaseClassPath="Communication"
      />
    </InterfaceClass>
   </InterfaceClass>
 </InterfaceClassLib>
</CAEXFile>
```

Bibliography

IEC 60027 (all parts), *Letter symbols to be used in electrical technology*

IEC 62264-1, *Enterprise-control system integration – Part 1: Models and terminology*

IEC 62714-2, *Engineering data exchange format for use in industrial automation systems engineering – Automation Markup Language – Part 2: Role class libraries*[2]

ISO 80000-1, *Quantities and units – Part 1: General*

2 To be published.

SOMMAIRE

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

FORMAT D'ÉCHANGE DE DONNÉES TECHNIQUES POUR UNE UTILISATION DANS L'INGÉNIERIE DES SYSTÈMES D'AUTOMATISATION INDUSTRIELLE – AUTOMATION MARKUP LANGUAGE –

Partie 1: Architecture et exigences générales

AVANT-PROPOS

1) La Commission Electrotechnique Internationale (CEI) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de la CEI). La CEI a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. A cet effet, la CEI – entre autres activités – publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de la CEI"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec la CEI, participent également aux travaux. La CEI collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
2) Les décisions ou accords officiels de la CEI concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de la CEI intéressés sont représentés dans chaque comité d'études.
3) Les Publications de la CEI se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de la CEI. Tous les efforts raisonnables sont entrepris afin que la CEI s'assure de l'exactitude du contenu technique de ses publications; la CEI ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de la CEI s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de la CEI dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de la CEI et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) La CEI elle-même ne fournit aucune attestation de conformité. Des organismes de certification indépendants fournissent des services d'évaluation de conformité et, dans certains secteurs, accèdent aux marques de conformité de la CEI. La CEI n'est responsable d'aucun des services effectués par les organismes de certification indépendants.
6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
7) Aucune responsabilité ne doit être imputée à la CEI, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de la CEI, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de la CEI ou de toute autre Publication de la CEI, ou au crédit qui lui est accordé.
8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.
9) L'attention est attirée sur le fait que certains des éléments de la présente Publication de la CEI peuvent faire l'objet de droits de brevet. La CEI ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de brevets et de ne pas avoir signalé leur existence.

La Norme internationale CEI 62714-1 a été établie par le sous-comité 65E: Dispositifs et leur intégration dans les systèmes de l'entreprise, du comité d'études 65 de la CEI: Mesure, commande et automation dans les processus industriels.

Le texte de cette norme est issu des documents suivants:

FDIS	Rapport de vote
65E/385/FDIS	65E/396/RVD

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à l'approbation de cette norme.

Cette publication a été rédigée selon les Directives ISO/CEI, Partie 2.

Une liste de toutes les parties de la série CEI 62714, publiées sous le titre général *Format d'échange de données techniques pour une utilisation dans l'ingénierie des systèmes d'automatisation industrielle – Automation Markup Language,* peut être consultée sur le site web de la CEI.

Le comité a décidé que le contenu de cette publication ne sera pas modifié avant la date de stabilité indiquée sur le site web de la CEI sous “http://webstore.iec.ch” dans les données relatives à la publication recherchée. A cette date, la publication sera

- reconduite;
- supprimée;
- remplacée par une édition révisée, ou
- amendée.

IMPORTANT – Le logo *"colour inside"* qui se trouve sur la page de couverture de cette publication indique qu'elle contient des couleurs qui sont considérées comme utiles à une bonne compréhension de son contenu. Les utilisateurs devraient, par conséquent, imprimer cette publication en utilisant une imprimante couleur.

INTRODUCTION

La CEI 62714 constitue une approche de l'échange de données qui cible le domaine de l'ingénierie de l'automatisation.

Le format d'échange de données défini dans la série CEI 62714 (Automation Markup Language, AML) est un format de données de type schéma XML mis au point afin de venir à l'appui de l'échange de données dans un environnement d'outils techniques hétérogène.

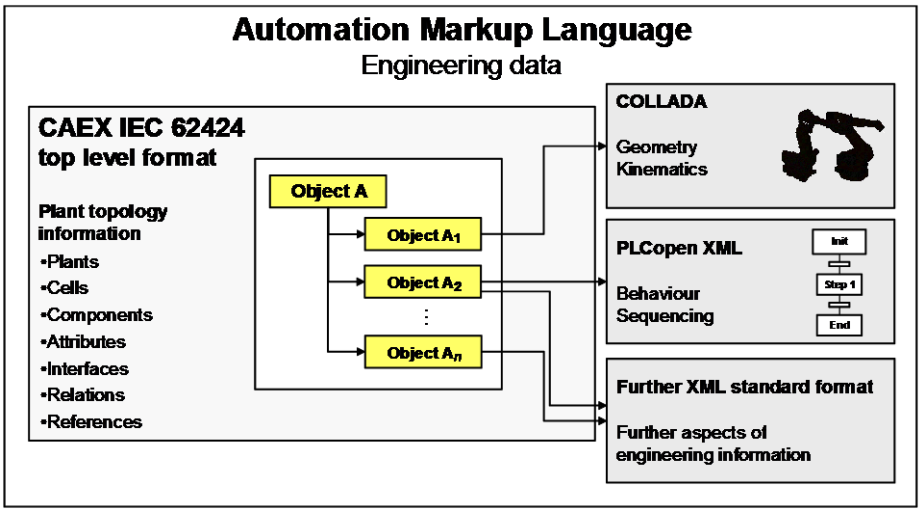
L'objectif de l'AML est l'interconnexion des outils techniques dans leurs différentes disciplines, par exemple, ingénierie des installations mécaniques, études d'électricité, ingénierie de procédés, ingénierie de commande de processus, développement des IHM, programmation PLC, programmation de robots, etc.

AML archive les informations techniques en respectant le paradigme orienté objet et permet la modélisation des composants d'installations physiques et logiques sous forme d'objets de données qui englobent différents aspects. Un objet peut comporter d'autres sous-objets, et peut lui-même faire partie intégrante d'une composition ou d'une agrégation plus importante. Les objets typiques que l'on trouve dans l'automatisation d'installations comprennent les informations concernant la topologie, la géométrie, la cinématique et la logique, tandis que la logique comprend pour sa part le séquencement, le comportement et la commande. Par conséquent, un objectif important de l'échange de données en ingénierie est l'échange de structures de données orientées objet, ainsi que la géométrie, la cinématique et la logique.

AML combine les formats de données industrielles existants, conçus pour l'archivage et l'échange de différents aspects des informations techniques. Ces formats de données sont utilisés «en l'état» dans le cadre de leurs propres spécifications et ne sont pas associés aux besoins du langage AML.

La caractéristique centrale de l'AML est le format de données central CAEX qui connecte les différents formats de données. Le langage AML a par conséquent une architecture de document répartie intrinsèque.

La Figure 1 illustre l'architecture AML de base et la répartition des informations concernant la topologie, la géométrie, la cinématique et la logique.



Légende

Anglais	Français
Engineering data	Données techniques
CAEX IEC 62424 top level format	Format central CAEX défini dans la CEI 62424

Anglais	Français
Object	objet
Plant topology information	Informations concernant la topologie de l'installation
Plants	Installations
Cells	Cellules
Components	Composants
Attributes	Attributs
Interfaces	Interfaces
References	Références
Geometry	Géométrie
Kinematics	Cinématique
Behaviour	Comportement
Sequencing	Séquencement
Init	Début
Step	Etape
End	Fin
Further XML standard format	Autre format standard XML
Further aspects of engineering information	Autres aspects des informations techniques

Figure 1 – Vue d'ensemble du format d'échange de données techniques (AML)

Du fait des différents aspects d'AML, la série CEI 62714 comporte différentes parties concentrées sur différents aspects:

- CEI 62714-1: Architecture et exigences générales

  Cette partie spécifie l'architecture AML générale, et la modélisation des données techniques, classes, instances, relations, références, hiérarchies, bibliothèques AML de base et concepts AML étendus. Elle constitue la norme de référence de toutes les parties futures, et fournit des mécanismes de référencement d'autres sous-formats.

- CEI 62714-2: Bibliothèques de classe de rôles

  Cette partie spécifie d'autres bibliothèques AML.

- CEI 62714-3: Géométrie et cinématique

  Cette partie spécifie la modélisation des informations concernant la géométrie et la cinématique.

- CEI 62714-4: Logique

  Cette partie spécifie la modélisation des informations relatives à la logique, au séquencement, au comportement et à la commande.

D'autres parties pourront être ajoutées à l'avenir afin d'interconnecter d'autres normes de données avec l'AML.

Tant qu'aucune autre partie ne décrit l'intégration d'autres normes, il est important de cibler un ensemble limité de formats de sous-données. A défaut, cela ouvrirait la voie à l'utilisation de tout format de données et l'échange de données ne fonctionnerait pas.

L'Annexe A fournit une introduction informative, des cas d'utilisation et des exemples d'AML.

L'Annexe B donne une représentation XML informative des bibliothèques définies dans la présente partie de la CEI 62714.

FORMAT D'ÉCHANGE DE DONNÉES TECHNIQUES POUR UNE UTILISATION DANS L'INGENIERIE DES SYSTÈMES D'AUTOMATISATION INDUSTRIELLE – AUTOMATION MARKUP LANGUAGE –

Partie 1: Architecture et exigences générales

1 Domaine d'application

La présente partie de la CEI 62714 spécifie les exigences générales et l'architecture du langage AML pour la modélisation des informations techniques échangées entre les outils techniques d'automatisation industrielle et des systèmes de commande. Ses dispositions s'appliquent aux fonctions exportation/importation des outils associés.

La présente partie de la CEI 62714 ne définit pas les détails de la procédure d'échange de données ou des exigences de mise en œuvre pour les outils d'importation/exportation.

2 Références normatives

Les documents suivants sont cités en référence de manière normative, en intégralité ou en partie, dans le présent document et sont indispensables pour son application. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

CEI 62424:2008, *Représentation de l'ingénierie de commande de processus – Demandes sous forme de diagrammes P&I et échange de données entre outils P&ID et outils PCE-CAE*

CEI 62714 (toutes les parties)*, Format d'échange de données techniques pour une utilisation dans l'ingénierie des systèmes d'automatisation industrielle – Automation Markup Language*

ISO/CEI 9834-8, T*echnologies de l'information – Interconnexion de systèmes ouverts (OSI) – Procédures opérationnelles pour les organismes d'enregistrement de l'OSI: Génération et enregistrement des identificateurs uniques universels (UUID) et utilisation de ces identificateurs comme composants d'identificateurs d'objets ASN.1*

ISO/PAS 17506, *Systèmes d'automatisation industrielle et intégration – Spécifications du schéma des actifs numériques COLLADA pour la visualisation 3D des données industrielles*

COLLADA 1.4.1:March 2008, COLLADA – Digital Asset Schema Release 1.4.1
(disponible sous <http://www.khronos.org/files/collada_spec_1_4.pdf>) (disponible en anglais seulement)

Extensible Markup Language (XML) 1.0 1.0:2004, W3C Recommendation
(disponible sous <http://www.w3.org/TR/2004/REC-xml-20040204/>) (disponible en anglais seulement)

PLCopen XML 2.0:December 3rd 2008 and *PLCopen XML* 2.0.1:May 8th 2009, XML formats for *IEC 61131*-3
(disponible sous <*http://www.plcopen.org>/)* (disponible en anglais seulement)

3 Termes, définitions et abréviations

3.1 Termes et définitions

Pour les besoins du présent document, les termes et définitions suivants s'appliquent.

3.1.1
AML
format d'échange de données XML pour les données d'ingénierie d'usine conformément à la CEI 62714

3.1.2
objet d'automatisation
entité physique ou logique du système automatisé

Note 1 à l'article: Un exemple d'objet d'automatisation est un composant d'automatisation, une vanne ou un signal.

3.1.3
objet AML
représentation de données d'un objet d'automatisation en relation avec une classe de rôle AML

Note 1 à l'article: Les objets AML sont les éléments centraux du langage AML. Ils représentent des instances et peuvent contenir des éléments d'administration, des attributs, des interfaces, des relations et des références.

3.1.4
classe AML
type d'objet AML prédéfini

Note 1 à l'article: Les classes AML sont archivées dans des bibliothèques AML.

Note 2 à l'article: Les classes AML définissent des solutions étalons réutilisables, caractérisées par des attributs, des interfaces et des objets regroupés.

Note 3 à l'article: Les classes AML peuvent être utilisées pour des instanciations multiples.

3.1.5
attribut AML
propriété d'un objet AML

Note 1 à l'article: Les attributs AML sont décrits comme élément XML correspondant à la CEI 62424:2008, A.2.4.

3.1.6
document AML
document CAEX spécifique suivant la CEI 62714, y compris tous les sous-documents référencés

Note 1 à l'article: Les documents AML peuvent être archivés sous forme de fichiers, mais également, par exemple, sous forme de trains de chaînes ou de données.

3.1.7
fichier AML
fichier CAEX spécifique suivant la CEI 62714-1, avec l'extension .aml, qui exclut tous les sous-fichiers référencés

3.1.8
interface AML
point de connexion unique qui appartient à un objet AML et qui peut être relié à une autre interface

Note 1 à l'article: Les interfaces permettent de décrire des relations entre les objets par la définition de InternalLinks CAEX. Les exemples d'interface AML sont une interface de signalisation, une interface de dispositif ou une interface de puissance.

3.1.9
bibliothèque AML
bibliothèque contenant les classes AML

3.1.10
port AML
objet AML qui représente le conteneur d'un groupe d'interfaces caractérisées par des propriétés supplémentaires

Note 1 à l'article: Les ports appartiennent à un objet AML parent et décrivent les interfaces complexes de cet objet. Les ports peuvent être connectés entre eux à un niveau d'abstraction plus élevé.

3.1.11
groupe AML
objet AML qui offre une certaine vue des objets de même nature

3.1.12
facette AML
objet AML qui offre une certaine vue des attributs ou des interfaces AML d'un objet de même nature spécifique

3.1.13
CAEX
format de données neutre basé sur le XML

Note 1 à l'article : CAEX est un format de données neutre conformément à la CEI 62424:2008, Article 7, Annexe A et Annexe C.

3.1.14
relation exemplaire-instance
relation entre l'instance et la classe correspondante, l'instance étant créée en copiant les structures de données de classe

Note 1 à l'article: L'instance reçoit un exemplaire de toutes les caractéristiques et propriétés de la classe AML source. Les modifications de la classe n'entraînent pas de modifications de l'instance. Les propriétés de classe de l'instance sont individualisées. La connaissance de la classe AML source permet de fournir d'autres exemplaires.

3.1.15
identifiant unique universel
UUID
identifiant unique des objets AML

Note 1 à l'article: L'abréviation "UUID" est dérivée du terme anglais développé correspondant "universal unique identifier".

3.1.16
identifiant unique global
GUID
mise en œuvre d'un UUID

Note 1 à l'article: Exemple réel de GUID: “{AC76BA86-7AD7-1033-7B44-A70000000000}”.

Note 2 à l'article: Dans la CEI 62714, les GUID sont également présentés sous forme abrégée telle que "GUID1", “GUID2” etc. Ceci permet la lisibilité et agit comme un GUID réel.

Note 3 à l'article: L'abréviation "GUID" est dérivée du terme anglais développé correspondant "global unique identifier".

3.1.17
relation d'héritage
relation entre deux classes AML

Note 1 à l'article: La classe dérivée hérite de tous les attributs et de toutes les caractéristiques de la classe parent.

3.1.18
instance
représentation des données d'un élément physique ou logique individuel

Note 1 à l'article: Les instances peuvent être étendues, par exemple, par des objets ou des attributs regroupés.

3.1.19
PropertySet
classe de rôle standard AML contenant un ensemble d'attributs prédéfinis de manière sémantique

3.1.20
topologie
structure hiérarchique d'un système, visualisable en tant qu'arborescence d'objets

Note 1 à l'article: Les hiérarchies multiples, les structures croisées et les réseaux d'objets sont inclus.

3.1.21
topologie de l'installation
structure hiérarchique d'une installation, visualisable en tant qu'arborescence d'objets

3.1.22
éditer, verbe
modéliser une structure de données d'un document externe destinée à être utilisée avec le format CAEX

Note 1 à l'article: Ceci permet de définir les relations entre les structures de données de documents externes indépendants.

3.1.23
relation
association d'objets CAEX

Note 1 à l'article: Des exemples de relations sont les relations parent-enfant et classe-instance.

3.1.24
liaison
connexion entre objets de type ExternalInterface CAEX

Note 1 à l'article: InternalLink CAEX permet de modéliser une liaison.

3.1.25
référence
association entre un objet InternalElement CAEX et les informations à archivage externe

3.2 Abréviations

Tableau 1 – Abréviations

AML	Langage de balisage d'automatisation (Automation Markup Language)
CAE	Ingénierie assistée par ordinateur (Computer Aided Engineering)
CAEX	Échange de données techniques assisté par ordinateur (en anglais Computer Aided Engineering eXchange)
COLLADA	Activité de conception coopérative (Collaborative design activity)

GUID	Identifiant unique global (Global unique identifier)
IHM	Interface homme-machine (Human machine interface
ID	Identifiant (Identifier)
MES	Système d'exécution de fabrication (Manufacturing execution system)
PLC	Automate programmable (Programmable logic controller)
URL	Adresse URL (Uniform resource locator)
URI	Identificateur de ressources uniformes (Uniform resource identifier)
UUID	Identifiant unique universel (Universal unique identifier)
XML	Langage de balisage extensible (Extensible Markup Language)

4 Conformité

Les exigences des Articles 5, 6, 7 et 8 doivent être satisfaites pour revendiquer la conformité avec la présente partie de la CEI 62714 eu égard à la prise en charge du langage AML.

5 Spécification de l'architecture AML

5.1 Généralités

La caractéristique centrale du langage AML est le format de données de base CAEX, un format de données neutre conformément à la CEI 62424:2008, Article 7, Annexe A et Annexe C, qui interconnecte les formats de données établis pour les aspects techniques des informations concernant la topologie, la géométrie, la cinématique, le comportement et le séquencement. Par conséquent, une caractéristique de base du langage AML est une architecture de document répartie intrinsèque qui cible les aspects techniques évoqués ci-dessus.

Les figures sont données uniquement pour illustration. La représentation graphique n'est pas normative.

5.2 Architecture AML générale

Les dispositions suivantes s'appliquent concernant l'architecture AML générale:

Informations concernant la topologie de l'installation: La topologie de l'installation agit comme la structure de données centrale des informations concernant l'ingénierie d'usine et doit être modélisée au moyen du format de données CAEX conformément à la CEI 62424:2008, Article 7, Annexe A et Annexe C. Les extensions sémantiques du format CAEX sont décrites séparément. Les structures à hiérarchies multiples et croisées doivent être utilisées au moyen du concept d'objet miroir conformément à la CEI 62424:2008, A.2.14. Les objets miroir ne doivent pas être modifiés; tous les changements doivent être opérés sur l'objet maître.

NOTE 1 Conformément à la CEI 62424:2008, A.2.14, un objet AML en relation avec un autre objet AML est appelé "objet miroir", l'objet AML connexe étant appelé "objet maître". L'objet miroir est considéré identique à l'objet maître. Ceci permet de placer une instance d'objet dans différentes hiérarchies de l'installation et permet ainsi la modélisation de réseaux d'objets complexes avec des structures croisées.

NOTE 2 La CEI 62714 ne modifie pas la syntaxe du format de données CAEX. Une vue d'ensemble informative et des exemples supplémentaires concernant la topologie de l'installation sont fournis en A.1.2 et à l'Annexe D de la CEI 62424:2008.

Informations concernant les références et les relations: Les références et les relations doivent être archivées selon 5.6 et 5.7. Les relations entre les informations à archivage externe doivent être archivées par des mécanismes CAEX. Le cas échéant, les partenaires de liaison associés doivent être indiqués dans la description de la topologie de l'installation

CAEX, au moyen des ExternalInterfaces CAEX. Ils doivent être déduits des classes d'interface standard AML spécifiées en 6.3.

NOTE 3 Les références illustrent les liaisons entre les objets CAEX et les informations à archivage externe. Une vue d'ensemble informative concernant les relations est fournie en A.1.5. Les références et l'édition des interfaces sont décrites dans les parties supplémentaires de la CEI 62714.

NOTE 4 Les relations illustrent les associations entre les objets CAEX.

Informations concernant la géométrie et la cinématique: Les informations relatives à la géométrie et à la cinématique doivent être archivées au moyen du format de données COLLADA™ [1]. Les interfaces COLLADA à interconnecter dans le format central doivent être éditées en tant que ExternalInterfaces CAEX.

NOTE 5 La CEI 62714 ne modifie pas la syntaxe du format de données COLLADA. Un exemple de présentation de la méthode de référencement du format COLLADA est fourni en A.1.3. Les détails sont à spécifier dans la CEI 62714-3.

NOTE 6 Les informations concernant la géométrie COLLADA de différents objets permettent de déduire de manière automatique une scène complète. Ces fichiers peuvent être référencés à partir du format CAEX et peuvent être interconnectés au moyen des mécanismes de liaison CAEX.

Informations concernant la logique: Les informations concernant la logique doivent être archivées au moyen du format de données XML PLCopen. Lorsque des éléments logiques, par exemple, des variables ou des signaux, sont à interconnecter dans le format central, ils doivent être édités en tant que ExternalInterfaces CAEX. Tous les éléments de format XML PLCopen qui sont édités dans le format central doivent avoir un identifiant unique au format XML PLCopen.

NOTE 7 Les informations concernant la logique décrivent des séquences d'actions et le comportement interne des objets, y compris les connexions E/S et les variables logiques. La CEI 62714 ne modifie pas le format XML PLCopen. Une vue d'ensemble informative de la méthode de référencement des informations concernant la logique est fournie en A.1.4. Les détails sont à spécifier dans la CEI 62714-4.

Référencement d'autres formats de données: L'extension future de la CEI 62714 est possible par la publication de parties supplémentaires spécifiant l'intégration d'autres formats de données XML qui utilisent les mécanismes de référence AML. Les détails peuvent être définis dans les parties supplémentaires de la CEI 62714.

Le format de données AML ne prévoit pas de contrôles de cohérence des contraintes, des valeurs d'attributs, des relations et références, ni l'exactitude sémantique des données contenues: ces éléments relèvent de la responsabilité de l'outil source ou cible, ou de l'application d'importation/exportation correspondante. Le langage AML permet uniquement la démonstration syntaxique du document par rapport aux schémas correspondants.

5.3 Versions de documents AML

Chaque document XML AML doit archiver la version AML sous-jacente que la présente norme respecte.

NOTE 1 Les dispositions normatives concernant les informations sur la version liées aux instances d'objets AML sont définies en 8.9. L'archivage des méta-informations spécifiques aux outils est défini en 5.4.

De fait, les dispositions suivantes s'appliquent:

- L'élément racine CAEX "CAEXFile" de chaque document central AML doit avoir l'élément enfant CAEX "AdditionalInformation".

---

1 COLLADA est la marque d'un produit fourni par Khronos Group. Cette information est donnée pour les besoins des utilisateurs de la présente norme et ne vaut pas approbation par la CEI du produit nommé. Des produits équivalents peuvent être utilisés s'ils peuvent être réputés donner les mêmes résultats.

- L'élément “AdditionalInformation” doit avoir un attribut “AutomationMLVersion”.
- La valeur de cet attribut «AutomationMLVersion» doit être archivée dans le document XML. Il doit être “2.0“ pour correspondre à la présente norme.
- Chaque document CAEX référencé doit suivre la même version AML du document racine. Le mélange de documents avec des versions AML différentes est explicitement interdit.
- Chaque document externe référencé doit également suivre les versions de schémas nommées, définies dans la spécification de versions AML ci-dessus. Le mélange de versions de documents externes hors d'une spécification de versions AML est explicitement interdit.

La Figure 2 illustre le texte XML pour un document CAEX suivant la version AML 2.0.

```
<CAEXFile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="
CAEX_ClassModel.xsd" FileName="AutomationMLStandardLibrary2010-01-14_v1.99.aml" SchemaVersion="2.15">
  <AdditionalInformation AutomationMLVersion="2.0"/>
```

Figure 2 – Informations concernant les versions de documents AML

- Chaque bibliothèque standard AML et chaque bibliothèque AML définie par l'utilisateur doivent préciser leur numéro de version en utilisant l'élément CAEX "Version". La syntaxe de la valeur du numéro de version n'est pas définie dans la présente partie de la CEI 62714.
- Le cas échéant, les classes CAEX doivent définir leur numéro de version en utilisant l'élément CAEX "Version". La syntaxe et la sémantique du numéro de version des classes dans une bibliothèque ne sont pas définies dans la présente partie de la CEI 62714.
- Il est interdit d'archiver les mêmes bibliothèques de différentes versions dans le même fichier AML.

  NOTE 2 Ceci assure le caractère unique des noms de bibliothèques AML dans un fichier AML.
- Le créateur d'un document AML doit s'assurer que seules les classes compatibles avec la version et les documents externes sont référencées.

La CEI 62714 repose sur les formats de documents suivants:

- CAEX version 2.15;
- PLCopenXML 2.0 et 2.0.1;
- COLLADA 1.5.0 conformément à l’ISO/PAS 17506 et COLLADA 1.4.1;
- Bibliothèques standard AML telles que spécifiées dans la présente partie de la CEI 62714 et les autres parties de la CEI 62714.

5.4 Méta-informations concernant l'outil source AML

Lorsqu'il est nécessaire de transférer les données définies par l'utilisateur d'un outil source vers un outil cible, il est nécessaire d'archiver les informations concernant l'outil source directement dans le document AML. De fait, les dispositions suivantes s'appliquent:

- Chaque document AML doit fournir des informations concernant l'outil qui a rédigé le document AML.
- Tous les outils qui participent à une chaîne d'outils d'échange de données doivent archiver ces informations dans le document CAEX de la même manière. De fait, le document peut contenir les informations concernant les différents outils d'une chaîne d'outils d'échange de données. Un outil peut supprimer les informations de rédaction d'autres outils. Ceci peut entraver l'échange de données itératives avec les autres outils: la suppression des informations de rédaction d’autres outils n'est de ce fait pas recommandée.
- Ces informations doivent être archivées comme partie intégrante de l'élément AdditionalInformation CAEX de l'objet racine du document CAEX.
- Le bloc AdditionalInfomation doit être nommé “WriterHeader”.

- Les méta-informations doivent fournir des renseignements concernant les éléments suivants:
  - le nom du logiciel d'exportation, de l'outil de rédaction
  - l'ID de l'outil de rédaction (il doit rester inchangé);
  - le fournisseur de l'outil de rédaction;
  - l'URL de l'outil de rédaction;
  - la version de produit de l'outil de rédaction;
  - le numéro de diffusion de produit de l'outil de rédaction;
  - le dernier temps d'écriture du rédacteur;
  - l'intitulé de projet du projet source;
  - l'ID de projet du projet source.
- Le contenu des méta-informations doit être défini par l'outil de rédaction et doit être du type xs:string.
- Les informations requises doivent être archivées en utilisant les attributs présentés dans le Tableau 2.

Tableau 2 – Méta-informations concernant l'outil source AML

Nom de balise XML	Type	Niveau	Exemple
WriterName	xs:string	Obligatoire	“ToolX AML Exporter”
WriterID	xs:string	Obligatoire	“ToolXToAML123”
WriterVendor	xs:string	Obligatoire	“ToolX Vendor”
WriterVendorURL	xs:string	Obligatoire	“http://www.ToolX-Vendor.org”
WriterVersion	xs:string	Obligatoire	“0.2”
WriterRelease	xs:string	Obligatoire	“123 prealpha”
LastWritingDateTime	xs:DateTime	Obligatoire	“2011-05-25T09:30:47”
WriterProjectTitle	xs:string	Facultatif	“eCarproduction”
WriterProjectID	xs:string	Facultatif	“eCarproduction_LinePLC.prj”

Pour la représentation XML des méta-informations, les dispositions suivantes s'appliquent:

- L'élément “WriterHeader” doit être un élément XML enfant de l'élément CAEX AdditionalInformation de l'élément racine de même nature.
- Chaque méta-information nommée dans le Tableau 2 doit être décrite comme un élément XML enfant de l'élément “WriterHeader”.
- Plusieurs méta-informations du même nom sont interdites dans le même élément “WriterHeader”.
- L'ordre des méta-informations doit être équivalent au Tableau 2.

La Figure 3 illustre le texte XML requis au moyen d'un exemple.

```
<AdditionalInformation>
  <WriterHeader>
    <WriterName>ToolX AML Exporter</WriterName>
    <ToolWriterID>ToolXToAML123</ToolWriterID>
    <WriterVendor>ToolX Vendor</WriterVendor>
    <WriterVendorURL>http://www.ToolX-Vendor.org</WriterVendorURL>
    <WriterVersion>0.1</WriterVersion>
    <WriterRelease>123 prealpha</WriterRelease>
    <LastWritingDateTime>2013-12-31T12:00:00</LastWritingDateTime>
    <WriterProjectTitle>eCarproduction</WriterProjectTitle>
    <WriterProjectID>eCarproduction_LinePLC.prj</WriterProjectID>
  </WriterHeader>
</AdditionalInformation>
```

Figure 3 – Texte XML des informations concernant l'outil source AML

5.5 Identification de l'objet

AML suit le paradigme orienté objet. Toutes les informations techniques sont modélisées comme objet ou appartiennent à un objet. Cependant, dans un environnement d'outils hétérogène, différents outils techniques utilisent des concepts différents pour l'identification des objets, par exemple, un nom, un identifiant ou un chemin unique. Certains outils autorisent les changements des identifiants au cours de la durée de vie, et d'autres pas. La CEI 62714 permet l'échange de données entre différents outils techniques avec ce type de concepts d'identification d'objet individuels. Du fait des caractéristiques décrites, la présente partie de la CEI 62714 neutralise cette variété et définit un concept d'identification d'objet obligatoire.

Les dispositions suivantes s'appliquent concernant l'identification d'objet:

- Conformément à la CEI 62424:2008, A.2.2.1, les classes AML (RoleClasses, InterfaceClasses et SystemUnitClasses) doivent être identifiées par leur balise "Name" CAEX.
- Ce nom doit être unique dans le niveau de hiérarchie de la bibliothèque AML correspondante au cours de la durée de vie de la classe.
- Conformément à la CEI 62424:2008, A.2.8, le référencement des classes doit être effectué par l'intermédiaire de chemins complets en utilisant les séparateurs de chemins correspondants.
- Toutes les instances d'objet AML (InternalElements CAEX et ExternalInterfaces CAEX) doivent être identifiées par leur balise "ID" CAEX. Cet identifiant doit être un identifiant unique universel (UUID) conformément à l'ISO/CEI 9834-8.

  NOTE 1 L'identifiant unique global (GUID) constitue une mise en œuvre possible de l'UUID.

  NOTE 2 Conformément à la CEI 62424:2008, A.3.15, la balise "ID" n'est pas obligatoire contrairement à la présente partie de la CEI 62714.

  NOTE 3 Dans la présente partie de la CEI 62714, les UUID sont présentés sous une forme abrégée telle que “GUID1”, “GUID2” etc.

  NOTE 4 La balise "Name" CAEX est un nom d'affichage; elle n'a qu'un caractère informatif et peut changer avec le temps ou en fonction de l'outil.
- Une fois créé, cet UUID ne doit jamais changer au cours de la durée de vie de l'objet correspondant de tous les outils participants.
- Les instances de référencement doivent utiliser la valeur "ID".
- Le référencement des interfaces CAEX doit être effectué en utilisant l'UUID correspondant de l'objet parent des interfaces, suivi de la chaîne de séparation “:” et du nom de l'instance d'interface.

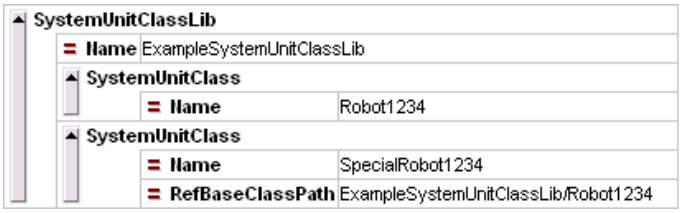
  EXEMPLE 1 “GUID:out”.
- Le référencement des attributs CAEX doit être effectué en utilisant l'UUID correspondant de l'objet parent des attributs, suivi de la chaîne de séparation “.” et du nom de l'attribut.

Si l'attribut est un attribut imbriqué, la chaîne de séparation est suivie du sous-chemin de l'attribut.

EXEMPLE 2 “GUID.Colour”.

EXEMPLE 3 “GUID.Colour.red”.

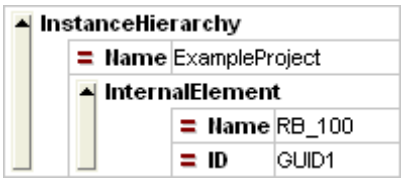
La Figure 4 donne un exemple de SystemUnitClassLib avec la SystemUnitClass “Robot1234” et une autre SystemUnitClass “SpecialRobot1234” déduite de la classe “Robot1234”.




```
<SystemUnitClassLib Name="ExampleSystemUnitClassLib">
  <SystemUnitClass Name="Robot1234"/>
  <SystemUnitClass Name="SpecialRobot1234" RefBaseClassPath="ExampleSystemUnitClassLib/Robot1234"/>
</SystemUnitClassLib>
```

Figure 4 – Exemple d'identification d'objet d'une classe AML

La Figure 5 donne un exemple de InstanceHierarchy avec un objet “RB_100” dont l'ID unique est représenté par la chaîne "GUID1".




```
<InstanceHierarchy Name="ExampleProject">
  <InternalElement Name="RB_100" ID="GUID1"/>
</InstanceHierarchy>
```

Figure 5 – Exemple d'identification d'objet d'une instance d'objet AML

5.6 Spécification des relations AML

5.6.1 Généralités

L'orientation axée sur les objets rend nécessaire de définir un mécanisme qui permet de déterminer les objets en association les uns avec les autres. La présente partie de la CEI 62714 distingue deux mécanismes d'archivage de ces informations: les références et les relations. Le paragraphe 5.6 cible les relations, tandis que 5.7 cible les références. Une vue d’ensemble informative concernant les relations et les références est fournie en A.1.5.

5.6.2 Relations parent-enfant entre les objets AML

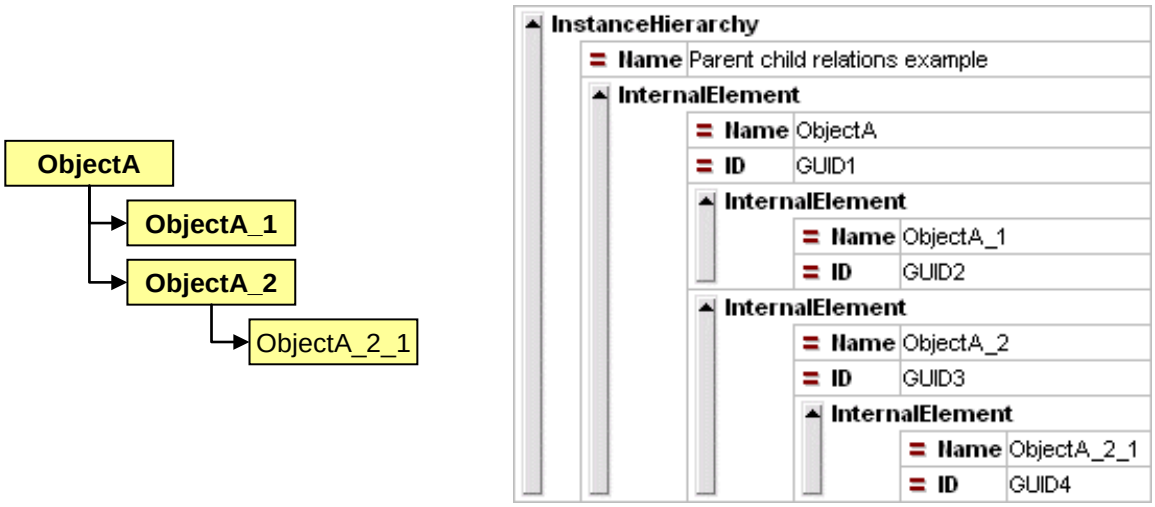
Les relations parent-enfant entre les instances d'objets permettent de représenter les structures d'objets hiérarchiques et de décrire un élément “consist-of-relation”.

La disposition suivante s'applique concernant les relations parent-enfant entre les objets AML:

- L'archivage des hiérarchies doit être effectué conformément à la CEI 62424:2008, Annexe A, par exemple, A.2.11.

  NOTE Outre les hiérarchies simples, il est également possible d'archiver les hiérarchies croisées (réseaux d'objets) conformément à la CEI 62424:2008, A.2.14.

La Figure 6 donne un exemple de hiérarchie d'objet et de son archivage.




```
<InstanceHierarchy Name="Parent child relations example">
    <InternalElement Name="ObjectA" ID="GUID1">
        <InternalElement Name="ObjectA_1" ID="GUID2"/>
        <InternalElement Name="ObjectA_2" ID="GUID3">
            <InternalElement Name="ObjectA_2_1" ID="GUID4"/>
        </InternalElement>
    </InternalElement>
</InstanceHierarchy>
```

Figure 6 – Exemple d'une relation parent-enfant entre objets AML

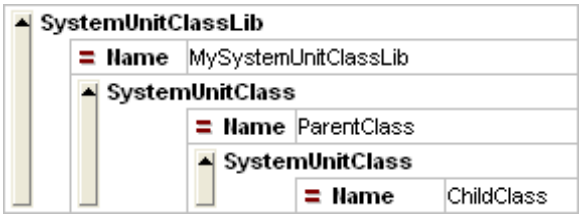
5.6.3 Relations parent-enfant entre les classes AML

Les dispositions suivantes s'appliquent concernant les relations parent-enfant entre les classes AML:

- Une relation parent-enfant entre les classes AML doit décrire leur seul voisinage hiérarchique. Ceci permet de définir toute structure hiérarchique définie par l'utilisateur.
- Cette relation n'a pas d'autre sémantique.

NOTE Une relation parent-enfant n'implique pas de relation d'héritage. Les relations d'héritage sont modélisées explicitement comme décrit en 5.6.4.

La Figure 7 donne un exemple de relation parent-enfant entre les classes “ParentClass” et “ChildClass”. La classe “ChildClass” n'a pas de relation d'héritage avec sa classe parent.




```
<SystemUnitClassLib Name="MySystemUnitClassLib">
    <SystemUnitClass Name="ParentClass">
        <SystemUnitClass Name="ChildClass"/>
    </SystemUnitClass>
</SystemUnitClassLib>
```

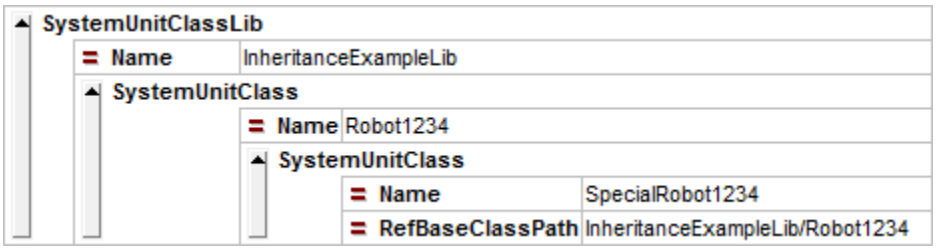
Figure 7 – Exemple d'une relation parent-enfant entre les classes

5.6.4 Relations d'héritage

Les dispositions suivantes s'appliquent concernant les relations d'héritage:

- L'héritage entre les classes doit être défini conformément à la CEI 62424:2008, A.2.7.
- Lorsque l'héritage est nécessaire, la classe parent doit être spécifiée à l'aide de la balise CAEX “RefBaseClassPath” comprenant le chemin complet de la classe conformément à la CEI 62424:2008, A.2.7.
- Si la classe parent souhaitée est placée à un niveau de hiérarchie au-dessus de la classe enfant, elle peut être spécifiée par l'archivage de son nom dans la balise CAEX “RefBaseClassPath” sans fournir le chemin complet.

La Figure 8 donne un exemple de la classe “Robot1234” et d'une autre classe “SpecialRobot1234” héritée de la classe “Robot1234”.




```
<SystemUnitClassLib Name="InheritanceExampleLib">
  <SystemUnitClass Name="Robot1234">
    <SystemUnitClass Name="SpecialRobot1234" RefBaseClassPath="
InheritanceExampleLib/Robot1234"/>
  </SystemUnitClass>
</SystemUnitClassLib>
```

Figure 8 – Exemple d'une relation d'héritage entre deux classes

Outre cet exemple, la balise CAEX “RefBaseClassPath” peut être la balise “InheritanceExampleLib/Robot1234” ou la balise “Robot1234”, dans la mesure où la classe parent est placée à un niveau de hiérarchie au-dessus de la classe “SpecialRobot1234”.

5.6.5 Relations classe-instance

Les instances sont caractérisées par un identifiant et un ensemble de paramètres uniques. Les dispositions suivantes s'appliquent:

- Un objet AML doit être modélisé comme InternalElement CAEX, en tant que partie intégrante de la InstanceHierarchy de même nature ou d'une SystemUnitClass.
- Un objet AML peut être un singleton sans relation avec toute SystemUnitClass.

  NOTE 1 Toutefois, un objet AML a une relation avec une classe de rôle AML standard.

  NOTE 2 Les instances sans aucune relation avec l'élément AutomationMLBaseRole sont possibles, mais sont des objets définis par l'utilisateur. Ce ne sont pas des objets AML.

- Si un objet AML a une relation classe-instance avec une SystemUnitClass, il doit être créé en tant qu'exemplaire de cette classe, y compris l'architecture interne de la classe et toutes les informations héritées.

  NOTE 3 Une SystemUnitClass est utilisée comme modèle de cette manière. Les changements opérés dans la SystemUnitClass ne se retrouvent pas automatiquement dans les objets "Automation" correspondants. Par ailleurs, l'objet "Automation" peut être transporté sans les informations de classe, il contient en lui-même l'ensemble des informations d'appartenance.

- L'extension ou la réduction des données d'instance par comparaison à la classe source est admise.

  NOTE 4 La classe source est destinée à être un point de départ approprié du modèle d'instance.

- La classe source reproduite doit être indiquée dans la balise CAEX “RefBaseSystemUnitPath” de l'instance en vue d'un usage ultérieur. Cette balise doit comporter le chemin et le nom complets de la classe source.

NOTE 5 Le changement de la classe source d'une instance n'implique pas un changement de l'instance. La mise à jour des instances est une fonctionnalité d'outil potentielle qui ne relève pas du domaine d'application de la présente partie de la CEI 62714.

- Il convient que les changements d'une classe source génèrent une nouvelle version de la classe avec un autre nom. Dans la nouvelle classe, il convient d'archiver le chemin complet de l'ancienne version de la classe dans la balise CAEX "Oldversion".

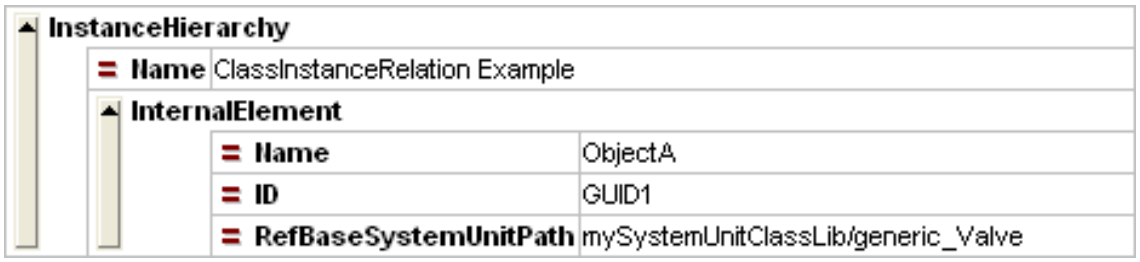
  NOTE 6 Cette disposition prend en charge le suivi des changements opérés dans les différentes versions d'une classe.

- L'héritage entre une SystemUnitClass et une instance d'objet n'est pas admis.

  NOTE 7 Une instance peut constituer uniquement un exemplaire de sa classe. Il s'agit d'une restriction par rapport à la CEI 62424:2008, A.2.7. L'héritage entre les classes et les instances peut faire partie intégrante d'une extension future.

- La relation entre une instance et une RoleClass doit être indiquée conformément à la CEI 62424:2008, A.2.7, par l'attribut “RefBaseRoleClassPath" de l'élément RoleRequirement d'appartenance. Contrairement à la CEI 62424:2008, A.2.7, aucun héritage n'est admis; toutes les spécifications de RoleClass doivent être reproduites dans l'objet AML correspondant.
- La relation entre une ExternalInterface CAEX et une InterfaceClass doit être indiquée conformément à la CEI 62424:2008, A.2.7. Contrairement à la CEI 62424, aucun héritage n'est admis; toutes les spécifications de InterfaceClass doivent être reproduites dans l'objet AML correspondant.

La Figure 9 donne un exemple de relation classe-instance entre l'objet “ObjectA” et une SystemUnitClass “generic_Valve” définie par l'utilisateur.




```
<InstanceHierarchy Name="ClassInstanceRelation Example">
  <InternalElement Name="ObjectA" ID="GUID1" RefBaseSystemUnitPath="mySystemUnitClassLib/generic_Valve"/>
</InstanceHierarchy>
```

Figure 9 – Exemple d'une relation classe-instance

Outre les dispositions standard concernant la relation classe-instance, la disposition spécifique suivante s'applique selon le concept miroir CAEX.

- La balise “RefBaseSystemUnitPath” peut indiquer une autre instance d'objet en lieu et place d'une SystemUnitClass selon le concept miroir de la CEI 62424:2008, A.2.14.

5.6.6 Relations entre instances

Les relations entre instances sont des relations entre deux interfaces des objets AML arbitraires.

Les dispositions suivantes s'appliquent concernant les relations entre instances:

- Les relations entre instances doivent être archivées conformément à la CEI 62424:2008, A.2.5.3 et A.2.14, grâce à la fonctionnalité InternalLink CAEX.
- Il convient d'archiver les InternalLinks CAEX dans le InternalElement de même nature qui est le parent commun le plus petit des objets CAEX connectés correspondants.

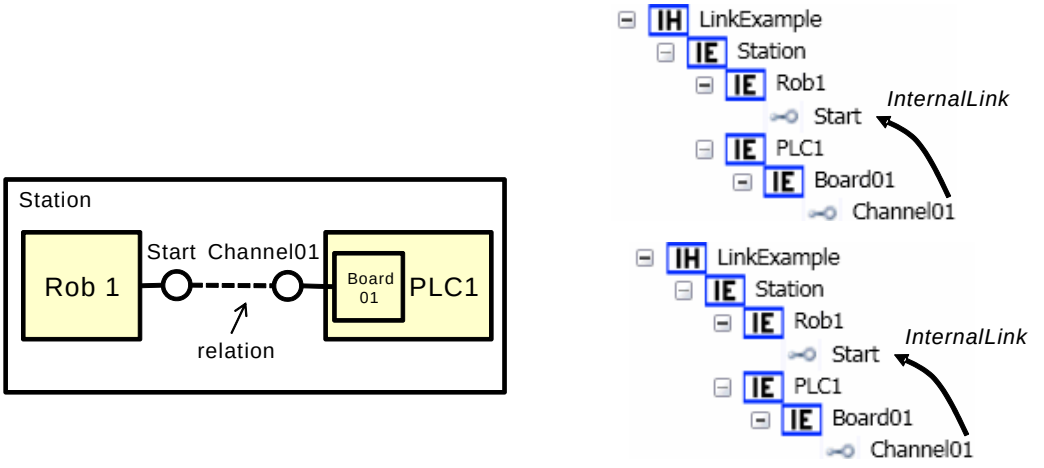
- Les relations entre instances doivent être définies uniquement entre les ExternalInterfaces CAEX qui appartiennent aux objets AML correspondants. Cette opération est conforme à la CEI 62424:2008, A.2.3.1.
- Il convient de déduire les ExternalInterfaces directement ou indirectement à partir d'une des classes d'interfaces standard AML.

  NOTE 1 La bibliothèque de classes d'interfaces standard AML est spécifiée en 6.3. Les classes d'interfaces définissent la sémantique de l'interface et donc la sémantique de la liaison. Une liaison entre interfaces sans référence à une classe d'interface n'a pas de sémantique.
- Les documents COLLADA peuvent être interconnectés. Les interfaces COLLADA correspondantes peuvent être tous les éléments ayant un URI valide. Si l'interconnexion de ces nœuds au format CAEX est nécessaire, lesdits nœuds doivent être édités dans ce format en ajoutant une ExternalInterface CAEX à l'objet correspondant. Cette ExternalInterface doit être déduite de la classe d'interface standard AML "COLLADAInterface" ou de l'une de ses dérivées.

  NOTE 2 La classe d'interface standard "COLLADAInterface" est spécifiée en 6.3.7. Les détails sont à spécifier dans la CEI 62714-3.
- Les documents XML PLCopen peuvent être interconnectés au moyen des interfaces de même nature correspondantes. Si l'interconnexion des éléments XML PLCopen au format CAEX est nécessaire, lesdits éléments doivent être édités en ajoutant une ExternalInterface CAEX à l'objet correspondant. Cette ExternalInterface doit être déduite de la classe d'interface standard AML "PLCopenXMLInterface" ou de l'une de ses dérivées.

  NOTE 3 La classe d'interface standard "PLCopenXMLInterface" est spécifiée en 6.3.8. Les détails sont à spécifier dans la CEI 62714-4.

La Figure 10a) décrit un exemple comprenant un robot "Rob1" et un automate PLC "PLC1", chacun d'entre eux ayant une interface de signal connectée. La Figure 10b) illustre cet exemple comme une hiérarchie d'objet.

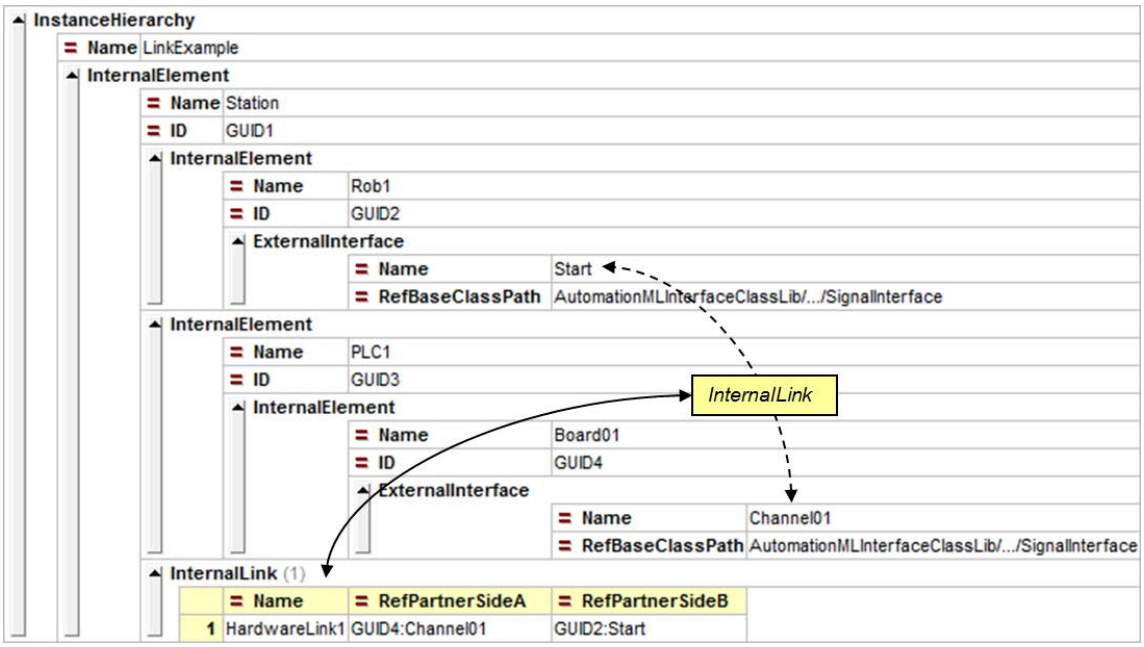


a) Relation en tant que schéma de principe **b) Relation en tant qu'arborescence d'objet**

Figure 10 – Exemple de relation en tant que schéma de principe et en tant qu'arborescence d'objet

La Figure 11 illustre la représentation AML de l'exemple indiqué. Le texte XML complet de l'élément InstanceHierarchy pour cet exemple, comprenant tous les objets AML "Station", "Rob1", "PLC1" et "Board01", y compris leurs interfaces, est présenté ci-dessous.

NOTE 4 L'expression "/…/" permet de réduire les chaînes de chemins données dans cet exemple afin d'améliorer la lisibilité.




```
<InstanceHierarchy Name="LinkExample">
  <InternalElement Name="Station" ID="GUID1">
    <InternalElement Name="Rob1" ID="GUID2">
      <ExternalInterface Name="Start" RefBaseClassPath="AutomationMLInterfaceClassLib/.../ SignalInterface">
        <Attribute Name="Type">
          <Value>digital</Value>
        </Attribute>
        <Attribute Name="Direction">
          <Value>In</Value>
        </Attribute>
      </ExternalInterface>
    </InternalElement>
    <InternalElement Name="PLC1" ID="GUID3">
      <InternalElement Name="Board01" ID="GUID4">
        <ExternalInterface Name="Channel01" RefBaseClassPath="AutomationMLInterfaceClassLib/.../ SignalInterface">
          <Attribute Name="Type">
            <Value>digital</Value>
          </Attribute>
          <Attribute Name="Direction">
            <Value>Out</Value>
          </Attribute>
        </ExternalInterface>
      </InternalElement>
    </InternalElement>
    <InternalLink Name="HardwareLink1" RefPartnerSideA="GUID4:Channel01" RefPartnerSideB="GUID2:Start"/>
  </InternalElement>
</InstanceHierarchy>
```

```
<InstanceHierarchy Name="LinkExample">
  <InternalElement Name="Station" ID="GUID1">
    <InternalElement Name="Rob1" ID="GUID2">
      <ExternalInterface Name="Start" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface"/>
    </InternalElement>
    <InternalElement Name="PLC1" ID="GUID3">
      <InternalElement Name="Board01" ID="GUID4">
        <ExternalInterface Name="Channel01" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterfac
      </InternalElement>
    </InternalElement>
    <InternalLink Name="HardwareLink1" RefPartnerSideA="GUID4:Channel01" RefPartnerSideB="GUID2:Start"/>
  </InternalElement>
</InstanceHierarchy>
```

Figure 11 – Exemple de relation entre les objets “PLC1” et “Rob1”

5.7 Spécification de référence de document AML

5.7.1 Généralités

Une référence de document permet la liaison entre un objet AML et un document externe pouvant contenir, par exemple, des informations concernant la géométrie, la cinématique ou la séquence. Le mécanisme de référence repose sur l'interface AML standard “ExternalDataConnector” ou l'une de ses dérivées.

5.7.2 Référencement de documents COLLADA

Le référencement des documents COLLADA doit être effectué sur la base de la classe d'interface standard AML “COLLADAInterface” ou de l'une de ses dérivées. Cette classe est spécifiée en 6.3.7. Les détails sont à spécifier dans la CEI 62714-3.

5.7.3 Référencement de documents XML PLCopen

Le référencement des documents XML PLCopen doit être effectué sur la base de l'interface standard AML “PLCopenXMLInterface” ou de l'une de ses dérivées. Cette classe est spécifiée en 6.3.8. Les détails sont à spécifier dans la CEI 62714-4.

5.7.4 Référencement de documents supplémentaires

Les extensions futures de la CEI 62714 peuvent ajouter des types d'interface supplémentaires pour le référencement de types de document supplémentaires. Elles sont spécifiées dans des parties distinctes de la CEI 62714 et non dans le domaine d'application de la présente norme. Les dispositions suivantes s'appliquent concernant ces extensions:

- Si des types de document supplémentaires sont à ajouter à la CEI 62714, ils doivent être modélisés avec des classes d'interfaces supplémentaires.
- Ces interfaces supplémentaires doivent être modélisées comme extension de la bibliothèque InterfaceClass AML et doivent être déduites directement ou indirectement de la classe d’interface standard ExternalDataConnector.
- Il convient d'archiver les références en utilisant les mêmes attributs standard fournis par les classes d'interface standard.
- La classe d’interface standard “ExternalDataConnector” doit être utilisée uniquement pour les types de document inclus dans la CEI 62714.

6 Bibliothèques de type AML

6.1 Généralités

L’Article 6 définit les bibliothèques de type AML essentielles avec les classes de même nature nécessaires pour la modélisation des concepts AML centraux. Tous les attributs décrits font partie intégrante de la bibliothèque standard AML et peuvent être supprimés de InstanceHierarchy lorsqu'ils ne sont pas nécessaires.

NOTE Les bibliothèques spécifiques au domaine relèvent du domaine d'application des autres parties de la CEI 62714.

6.2 Dispositions générales

Les dispositions suivantes s’appliquent concernant les bibliothèques de type AML:

- Tous les objets AML doivent être associés directement ou indirectement à la classe de rôle “AutomationMLBaseRole”.
- Toutes les interfaces doivent être associées directement ou indirectement à une classe d'interface AML.

- Les attributs AML doivent être utilisés le cas échéant et peuvent être supprimés des objets AML lorsqu'ils ne sont pas nécessaires.

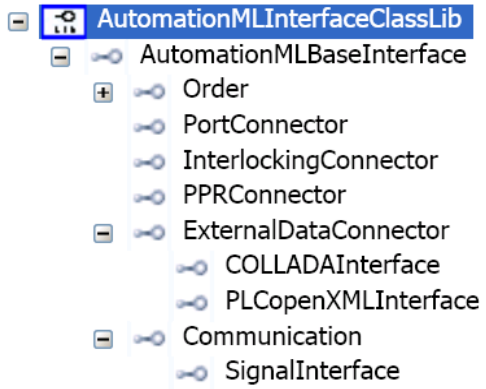
6.3 Bibliothèque de classes d'interface AML – AutomationMLInterfaceClassLib

6.3.1 Généralités

La bibliothèque AutomationMLInterfaceClassLib suivante est modélisée conformément à la CEI 62424:2008, Article 7, Annexe A et Annexe C. La CEI 62714 utilise le concept d'interface CAEX. Les extensions définies par l'utilisateur de cette bibliothèque AML sont admises comme cela est spécifié en 7.3.

Chaque interface doit être déduite directement ou indirectement d'une classe de la bibliothèque AutomationMLInterfaceClassLib standard suivante, selon le Tableau 3. Les paragraphes 6.3.2 à 6.3.10 spécifient les classes d'interfaces de manière détaillée.

Tableau 3 – Classes d'interfaces de la bibliothèque AutomationMLInterfaceClassLib

Bibliothèque des classes InterfaceClass AML	InterfaceClass	Description
	AutomationMLBaseInterface	Type d'interface abstraite
	Order	Interface de description des classements
	PortConnector	Interface de description des ports
	PPRConnector	Connecteur utilisé pour l'interconnexion des produits, ressources ou processus
	ExternalDataConnector	Interface de connecteurs générique avec les données externes
	COLLADAInterface	Interface avec un document COLLADA
	PLCopenXMLInterface	Interface avec un document XML PLCopen
	Communication	Interface de communication générique
	SignalInterface	Interface de signal générique

La Figure 12 présente une vue de tableau et la Figure 13 présente le texte XML de la bibliothèque InterfaceClassLib de type AML standard. Les paragraphes 6.3.2 à 6.3.10 fournissent des informations détaillées concernant les classes.

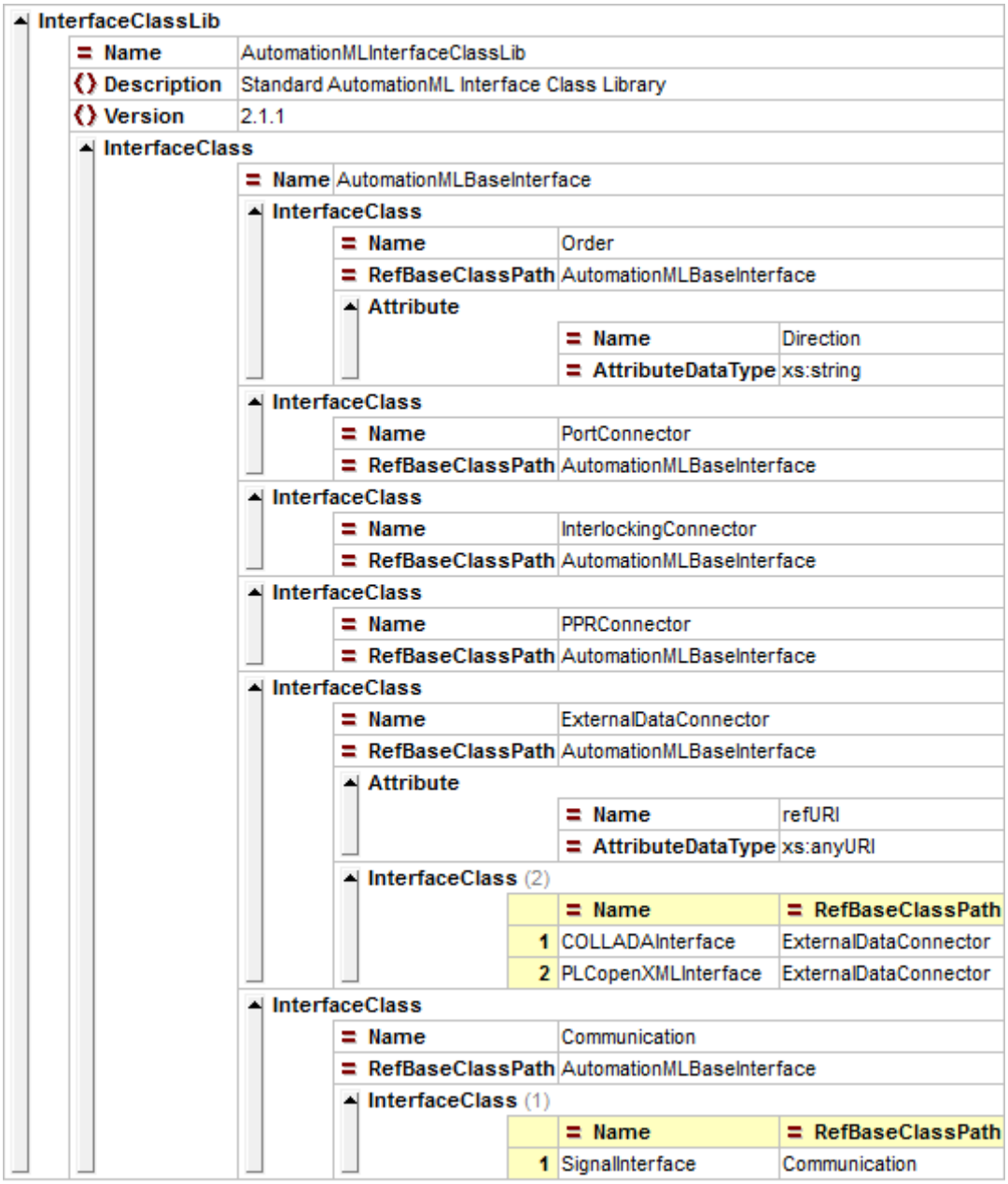


Figure 12 – Bibliothèque de classes d'interfaces de type AML

```
<InterfaceClassLib Name="AutomationMLInterfaceClassLib">
  <Description>Standard AutomationML Interface Class Library</Description>
  <Version>2.1.1</Version>
  <InterfaceClass Name="AutomationMLBaseInterface">
    <InterfaceClass Name="Order" RefBaseClassPath="AutomationMLBaseInterface">
      <Attribute Name="Direction" AttributeDataType="xs:string"/>
    </InterfaceClass>
    <InterfaceClass Name="PortConnector" RefBaseClassPath="AutomationMLBaseInterface"/>
    <InterfaceClass Name="InterlockingConnector" RefBaseClassPath="AutomationMLBaseInterface"/>
    <InterfaceClass Name="PPRConnector" RefBaseClassPath="AutomationMLBaseInterface"/>
    <InterfaceClass Name="ExternalDataConnector" RefBaseClassPath="AutomationMLBaseInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI"/>
      <InterfaceClass Name="COLLADAInterface" RefBaseClassPath="ExternalDataConnector"/>
      <InterfaceClass Name="PLCopenXMLInterface" RefBaseClassPath="ExternalDataConnector"/>
    </InterfaceClass>
    <InterfaceClass Name="Communication" RefBaseClassPath="AutomationMLBaseInterface">
      <InterfaceClass Name="SignalInterface" RefBaseClassPath="Communication"/>
    </InterfaceClass>
  </InterfaceClass>
</InterfaceClassLib>
```

Figure 13 – Description XML de la bibliothèque de classes d'interfaces de type AML

6.3.2 Bibliothèque InterfaceClass AutomationMLBaseInterface

Le Tableau 4 spécifie la classe d'interface "AutomationMLBaseInterface".

Tableau 4 – Bibliothèque InterfaceClass AutomationMLBaseInterface

Nom de classe	AutomationMLBaseInterface	
Description	La classe d'interface "AutomationMLBaseInterface" est un type d'interface abstraite de base et doit être utilisée comme classe parent pour la description de toutes les classes d'interfaces AML.	
Classe parent	Aucun	
Attributs	Aucun	

6.3.3 InterfaceClass Order

Le Tableau 5 spécifie la classe d'interface "Order".

Tableau 5 – InterfaceClass "Order"

Nom de classe	Order	
Description	La classe d'interface "Order" est une classe abstraite qui doit être utilisée pour la description des classements, par exemple un successeur ou un prédécesseur.	
Classe parent	AutomationMLInterfaceClassLib/AutomationMLBaseInterface	
Attributs	Direction (type="xs:string")	L'attribut "Direction" doit permettre de spécifier la direction. Les valeurs admises sont "In", "Out" ou "InOut".

6.3.4 InterfaceClass PortConnector

Le Tableau 6 spécifie la classe d'interface "PortConnector".

Tableau 6 – InterfaceClass PortConnector

Nom de classe	PortConnector	
Description	La classe d'interface "PortConnector" doit permettre d'assurer une relation de haut niveau entre les ports. Une vue d'ensemble du concept Port est fournie en A.2.2.	
Classe parent	AutomationMLInterfaceClassLib/AutomationMLBaseInterface	
Attributs	Aucun	

6.3.5 InterfaceClass PPRConnector

Le Tableau 7 spécifie la classe d'interface "PPRConnector".

Tableau 7 – InterfaceClass PPRConnector

Nom de classe	PPRConnector	
Description	La classe d'interface "PPRConnector" doit permettre d'assurer une relation entre les ressources, les produits et les processus. Voir A.2.6 pour de plus amples informations.	
Classe parent	AutomationMLInterfaceClassLib/AutomationMLBaseInterface	
Attributs	Aucun	

6.3.6 InterfaceClass ExternalDataConnector

Le Tableau 8 spécifie la classe d'interface "ExternalDataConnector".

Tableau 8 – InterfaceClass ExternalDataConnector

Nom de classe	ExternalDataConnector	
Description	La classe d'interface "ExternalDataConnector" est un type d'interface abstraite de base et doit être utilisée pour la description des interfaces de connecteurs de référencement des documents externes. Les classes "COLLADAInterface" et "PLCopenXMLInterface" sont déduites de cette classe. Toutes les classes de connecteurs existantes et futures doivent être déduites directement ou indirectement de cette classe.	
Classe parent	AutomationMLInterfaceClassLib/AutomationMLBaseInterface	
Attribut	refURI (type="xs:anyURI")	L'attribut "refURI" doit servir à l'archivage du chemin vers le document externe de référence.

6.3.7 InterfaceClass COLLADAInterface

Le Tableau 9 spécifie la classe d'interface “COLLADAInterface”. Les détails sont à spécifier dans la CEI 62714-3.

Tableau 9 – InterfaceClass COLLADAInterface

Nom de classe	COLLADAInterface	
Description	La classe d'interface “COLLADAInterface” doit permettre de référencer les documents COLLADA externes et d'éditer des interfaces définies dans un document COLLADA externe. Les détails sont à spécifier dans la CEI 62714-3.	
Classe parent	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector	
Attributs	Aucun	

6.3.8 InterfaceClass PLCopenXMLInterface

Le Tableau 10 spécifie la classe d'interface “PLCopenXMLInterface”. Les détails sont à spécifier dans la CEI 62714-4.

Tableau 10 – InterfaceClass PLCopenXMLInterface

Nom de classe	PLCopenXMLInterface	
Description	La classe d'interface “PLCopenXMLInterface” doit permettre de référencer les documents XML PLCopen externes ou d'éditer des signaux ou des variables définis dans le cadre d'une description logique XML PLCopen. Les détails sont à spécifier dans la CEI 62714-4.	
Classe parent	AutomationMLBaseInterface/ExternalDataConnector	
Attributs	Aucun	

6.3.9 InterfaceClass Communication

Le Tableau 11 spécifie la classe d'interface “Communication”.

Tableau 11 – InterfaceClass Communication

Nom de classe	Communication	
Description	La classe d'interface “Communication" est un type d'interface abstraite de base et doit être utilisée pour la description des interfaces liées à la communication. Les autres classes liées à la communication doivent être déduites directement ou indirectement de cette classe.	
Classe parent	AutomationMLInterfaceClassLib/AutomationMLBaseInterface	
Attributs	Aucun	

6.3.10 InterfaceClass SignalInterface

Le Tableau 12 spécifie la classe d’interface “SignalInterface”.

Tableau 12 – InterfaceClass SignalInterface

Nom de classe	SignalInterface	
Description	La classe d'interface “SignalInterface” doit permettre de modéliser les signaux. Ce type d'interface est configurable et permet de décrire des entrées et des sorties numériques et analogiques, ainsi que des combinaisons entrées-sorties configurables. Un exemple est décrit à la Figure 10.	
Classe parent	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication	
Attributs	Aucun	

6.4 Bibliothèque de classes de rôles de type AML – AutomationMLBaseRoleClassLib

6.4.1 Généralités

Le paragraphe 6.4 définit une bibliothèque type AML des classes de rôles standard essentielles pour la modélisation des concepts AML centraux. Un rôle est une classe qui décrit une fonctionnalité abstraite sans définir la mise en œuvre technique sous-jacente. Une “Ressource” ou un “Robot” constitue un exemple de classes de rôles. Tout en associant une classe de rôle à un objet AML, cet objet AML comporte une sémantique. Les bibliothèques étendues supplémentaires sont à décrire dans la CEI 62714-2. Tous les attributs décrits font partie intégrante de la bibliothèque standard AML et peuvent être supprimés de InstanceHierarchy lorsqu'ils ne sont pas nécessaires.

Chaque objet AML et chaque classe de rôle définie par l'utilisateur doivent avoir une référence directe ou indirecte à l'un des rôles dans cette bibliothèque AML. Si un certain rôle est trop spécifique, il convient de référencer le rôle parent suivant. Les Figure 14 à 16 présentent la RoleClass de base standard comme arborescence d'objet, table XML et texte XML. Les détails de chaque classe sont donnés de 6.4.2 à 6.4.13.

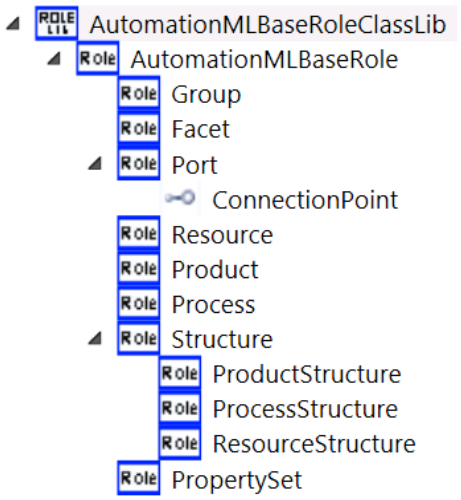


Figure 14 – Bibliothèque de classes de rôles de type AML

RoleClassLib

Name	AutomationMLBaseRoleClassLib
Description	AutomationML base role library
Version	2.1.1

RoleClass

Name	AutomationMLBaseRole

RoleClass

Name	Group
RefBaseClassPath	AutomationMLBaseRole

Attribute (1)

	Name	AttributeDataType
1	AssociatedFacet	xs:string

RoleClass

Name	Facet
RefBaseClassPath	AutomationMLBaseRole

RoleClass

Name	Port
RefBaseClassPath	AutomationMLBaseRole

Attribute (3)

	Name	AttributeDataType	Attribute
1	Direction	xs:string	
2	Cardinality	xs:complexType	Attribute (2)
3	Category	xs:string	

Attribute (2)

	Name	AttributeDataType
1	MinOccur	xs:uint
2	MaxOccur	xs:uint

ExternalInterface

Name	ConnectionPoint
RefBaseClassPath	AutomationMLInterfaceClassLib@AutomationMLInterfaceClassLib/AutomationMLBaseInterface/PortConnector

RoleClass

Name	Resource
RefBaseClassPath	AutomationMLBaseRole

RoleClass

Name	Product
RefBaseClassPath	AutomationMLBaseRole

RoleClass

Name	Process
RefBaseClassPath	AutomationMLBaseRole

RoleClass

Name	Structure
RefBaseClassPath	AutomationMLBaseRole

RoleClass (3)

	Name	RefBaseClassPath
1	ProductStructure	AutomationMLBaseRole/Structure
2	ProcessStructure	AutomationMLBaseRole/Structure
3	ResourceStructure	AutomationMLBaseRole/Structure

RoleClass

Name	PropertySet
RefBaseClassPath	AutomationMLBaseRole

Figure 15 – AutomationMLBaseRoleClassLib

```
<RoleClassLib Name="AutomationMLBaseRoleClassLib">
  <Description>AutomationML base role library </Description>
  <Version>2.1.1</Version>
  <RoleClass Name="AutomationMLBaseRole">
    <RoleClass Name="Group" RefBaseClassPath="AutomationMLBaseRole">
      <Attribute Name="AssociatedFacet" AttributeDataType="xs:string"/>
    </RoleClass>
    <RoleClass Name="Facet" RefBaseClassPath="AutomationMLBaseRole"/>
    <RoleClass Name="Port" RefBaseClassPath="AutomationMLBaseRole">
      <Attribute Name="Direction" AttributeDataType="xs:string"/>
      <Attribute Name="Cardinality" AttributeDataType="xs:complexType">
        <Attribute Name="MinOccur" AttributeDataType="xs:uint"/>
        <Attribute Name="MaxOccur" AttributeDataType="xs:uint"/>
      </Attribute>
      <Attribute Name="Category" AttributeDataType="xs:string"/>
      <ExternalInterface Name="ConnectionPoint" RefBaseClassPath=
       "AutomationMLInterfaceClassLib@AutomationMLInterfaceClassLib/AutomationMLBaseInterface/PortConnector"/>
    </RoleClass>
    <RoleClass Name="Resource" RefBaseClassPath="AutomationMLBaseRole"/>
    <RoleClass Name="Product" RefBaseClassPath="AutomationMLBaseRole"/>
    <RoleClass Name="Process" RefBaseClassPath="AutomationMLBaseRole"/>
    <RoleClass Name="Structure" RefBaseClassPath="AutomationMLBaseRole">
      <RoleClass Name="ProductStructure" RefBaseClassPath="AutomationMLBaseRole/Structure"/>
      <RoleClass Name="ProcessStructure" RefBaseClassPath="AutomationMLBaseRole/Structure"/>
      <RoleClass Name="ResourceStructure" RefBaseClassPath="AutomationMLBaseRole/Structure"/>
    </RoleClass>
    <RoleClass Name="PropertySet" RefBaseClassPath="AutomationMLBaseRole"/>
  </RoleClass>
</RoleClassLib>
```

Figure 16 – Texte XML de la bibliothèque AutomationMLBaseRoleClassLib

6.4.2 RoleClass AutomationMLBaseRole

Le Tableau 13 spécifie la classe de rôle “AutomationMLBaseRole”.

Tableau 13 – RoleClass AutomationMLBaseRole

Nom de classe	AutomationMLBaseRole	
Description	La classe de rôle “AutomationMLBaseRole” est un type de rôle abstrait de base et constitue la classe de base pour toutes les classes de rôles standard ou définies par l'utilisateur.	
Classe parent	Aucun	
Attributs	Aucun	

6.4.3 RoleClass Group

Le Tableau 14 spécifie la classe de rôle “Group”.

Tableau 14 – RoleClass Group

Nom de classe	Group	
Description	La classe de rôle "Group" est un type de rôle pour les objets qui permettent de regrouper les objets miroirs associés d'un certain point de vue technique. Les objets "Group" AML doivent référencer ce rôle. Les détails et les exemples sont spécifiés en 8.4.	
Classe parent	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Attributs	“AssociatedFacet” (type=”xs:string”)	L'attribut “AssociatedFacet” doit être utilisé pour définir le nom de l'élément Facet correspondant. Exemple: AssociatedFacet = “PLCFacet”.

6.4.4 RoleClass Facet

Le Tableau 15 spécifie la classe de rôle “Facet”.

Tableau 15 – RoleClass Facet

Nom de classe	Facet	
Description	La classe de rôle "Facet" est un type de rôle pour les objets offrant une vision secondaire des attributs ou des interfaces d'un objet AML. Les objets "Facet" AML doivent référencer ce rôle. Les détails et les exemples sont spécifiés en 8.3.	
Classe parent	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Attributs	Aucun	

6.4.5 RoleClass Port

Le Tableau 16 spécifie la classe de rôle “Port”.

Tableau 16 – Attributs facultatifs des objets Port AML

Nom de classe	Port	
Description	La classe de rôle "Port" est un type de rôle pour les objets qui regroupe de nombreuses interfaces et permet de décrire les interfaces complexes de cette manière. Les objets "Port" AML doivent référencer ce rôle. Les détails et les exemples sont spécifiés en 8.2.	
Classe parent	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Attributs	Direction (type=”xs:string”)	Cet attribut doit servir à décrire la direction du Port. La valeur utilisée doit être l'une des valeurs suivantes: “In”, “Out” ou “InOut”. Les ports dont la direction est “In” peuvent être connectés uniquement aux ports dont la direction est “Out” ou “InOut” et les ports dont la direction est “Out” peuvent être connectés uniquement aux ports dont la direction est “In” ou “InOut”. Les ports dont la direction est “InOut” peuvent être connectés aux ports dont la direction est arbitraire. Exemples: Direction = “Out” (par exemple une fiche) Direction = “In” (par exemple une prise femelle) Direction = “InOut” Cette information peut servir, par exemple, à démontrer la validité d'une connexion. NOTE La validité de ces connexions ne relève pas du domaine d'application de la CEI 62714, mais constitue en revanche une fonctionnalité d'outil.
	„Cardinality “	Cet attribut est un type d'attribut complexe et ne doit pas avoir de valeur. Les sous-attributs correspondants sont décrits dans le Tableau 17.
	“Category” (type=”xs:string”)	L'attribut “Category” décrit le type de port. La valeur de cet attribut est définie par l'utilisateur. Il est admis de connecter uniquement les ports ayant la même valeur de catégorie. Exemple: catégorie = “MaterialFlow”.

L'attribut “Cardinality” comprend deux sous-attributs décrits dans le Tableau 17.

Tableau 17 – Sous-attributs de l'attribut “Cardinality”

Attribut	Type	Description	Exemple
“MinOccur”	xs:unsignedInt	La valeur "MinOccur décrit le nombre potentiel minimum de connexions vers ou en provenance de ce port. Les valeurs de l'attribut doivent être supérieures ou égales à 0.	MinOccur = 1 Cela signifie qu'il convient de connecter ce port à au moins un autre port.
“MaxOccur”	xs:unsignedInt	La valeur "MaxOccur décrit le nombre potentiel maximum de connexions vers ou en provenance de ce port. Les valeurs de l'attribut doivent être supérieures ou égales à MinOccur, ou 0 qui désigne l'infini.	MaxOccur = 3 Cela signifie que ce port peut être connecté uniquement avec trois autres ports au maximum.

De plus, l'objet Port AML doit comporter un élément ExternalInterface CAEX déduit de l'InterfaceClass AML“PortConnector” (voir Tableau 18).

NOTE Cette interface permet de connecter le Port considéré avec de nombreux autres ports à un niveau abstrait, sans aucune description détaillée des relations internes entre les sous-interfaces (voir Figure A.13).

Tableau 18 – Interfaces de la classe Port AML

Interface	Type	Description	Exemple
Le nom est défini par l'utilisateur, par exemple, “ConnectionPoint”	PortConnector	Cette interface CAEX permet de connecter ce Port avec de nombreux autres ports à un niveau abstrait. Les relations internes entre les interfaces à port unique ne sont pas décrites de cette manière.	Voir A.2.2.2.

6.4.6 RoleClass Resource

Le Tableau 19 spécifie la classe de rôle “Resource”.

Tableau 19 – RoleClass Resource

Nom de classe	Resource	
Description	La classe de rôle “Resource” est un type de rôle abstrait de base et constitue la classe de base pour tous les rôles de ressource AML. Elle décrit les installations, matériels ou autres ressources de production. Les objets de ressource AML doivent référencer ce rôle directement ou indirectement. Des exemples sont spécifiés en A.2.6.	
Classe parent	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Attributs	Aucun	

De plus, le cas échéant, les objets Resource AML doivent comporter un élément ExternalInterface CAEX “PPRConnector” afin de créer les relations avec les produits et les processus (voir 6.3.5).

6.4.7 RoleClass Product

Le Tableau 20 spécifie la classe de rôle “Product".

Tableau 20 – RoleClass Product

Nom de classe	Product	
Description	La classe de rôle “Product” est un type de rôle abstrait de base et constitue la classe de base pour tous les rôles de produit AML. Elle décrit les produits, les parties de produits ou les matériaux liés aux produits qui sont transformés dans l'installation décrite. Les objets de produit AML doivent référencer ce rôle directement ou indirectement. Des exemples sont spécifiés en A.2.6.	
Classe parent	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Attributs	Aucun	

De plus, le cas échéant, les objets de produit AML doivent comporter un élément ExternalInterface CAEX “PPRConnector” afin de créer les relations avec les ressources et les processus (voir 6.3.5).

6.4.8 RoleClass Process

Le Tableau 21 spécifie la classe de rôle “Process”.

Tableau 21 – RoleClass Process

Nom de classe	Process	
Description	La classe de rôle “Process” est un type de rôle abstrait de base et constitue la classe de base pour tous les rôles de processus AML. Elle décrit les processus liés à la production. Les objets de processus AML doivent référencer ce rôle directement ou indirectement. Des exemples sont spécifiés en A.2.6.	
Classe parent	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Attributs	Aucun	

De plus, le cas échéant, les objets de processus AML doivent comporter un élément ExternalInterface CAEX “PPRConnector” afin de créer les relations avec les produits et les ressources (voir 6.3.5).

6.4.9 RoleClass Structure

Le Tableau 22 spécifie la classe de rôle “Structure".

Tableau 22 – RoleClass Structure

Nom de classe	Structure	
Description	La classe de rôle “Structure” est un type de rôle abstrait de base pour les objets utilisés comme éléments de structure dans la hiérarchie de l'installation, par exemple, un dossier, un site ou une chaîne de fabrication. Les objets de structure AML doivent référencer ce rôle directement ou indirectement.	
Classe parent	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Attributs	Aucun	

6.4.10 RoleClass ProductStructure

Le Tableau 23 spécifie la classe de rôle “ProductStructure”.

Tableau 23 – RoleClass ProductStructure

Nom de classe	ProductStructure	
Description	La classe de rôle “ProductStructure” est un type de rôle abstrait pour une hiérarchie d'objets orientés produit. Les objets de structure de produit AML doivent référencer ce rôle directement ou indirectement.	
Classe parent	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure	
Attributs	Aucun	

6.4.11 RoleClass ProcessStructure

Le Tableau 24 spécifie la classe de rôle “ProcessStructure”.

Tableau 24 – RoleClass ProcessStructure

Nom de classe	ProcessStructure	
Description	La classe de rôle “ProcessStructure” est un type de rôle abstrait pour une hiérarchie d'objets orientés processus. Les objets de structure de processus AML doivent référencer ce rôle directement ou indirectement.	
Classe parent	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure	
Attributs	Aucun	

6.4.12 RoleClass ResourceStructure

Le Tableau 25 spécifie la classe de rôle “ResourceStructure".

Tableau 25 – RoleClass ResourceStructure

Nom de classe	ResourceStructure	
Description	La classe de rôle “ResourceStructure” est un type de rôle abstrait pour une hiérarchie d'objets orientés ressource. Les objets de structure de ressource AML doivent référencer ce rôle directement ou indirectement.	
Classe parent	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure	
Attributs	Aucun	

6.4.13 RoleClass PropertySet

Le Tableau 26 spécifie la classe de rôle “PropertySet”.

Tableau 26 – RoleClass PropertySet

Nom de classe	PropertySet	
Description	La classe de rôle “PropertySet” est un type de rôle abstrait qui permet de définir des ensembles de propriétés correspondant à un certain aspect technique. Les objets Ensembles de propriétés AML doivent référencer ce rôle directement ou indirectement. Les dispositions normatives sont décrites en 8.5 et les détails et exemples y afférents sont spécifiés en A.2.5.	
Classe parent	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Attributs	Aucun	

7 Modélisation des données définies par l'utilisateur

7.1 Généralités

L’Article 7 décrit de quelle manière les données définies par l'utilisateur peuvent être modélisées en langage AML. La modélisation des données spécifiques définies par l'utilisateur constitue un concept central du langage AML. Les données définies par l'utilisateur sont les attributs, InterfaceClasses et RoleClasses CAEX qui ne sont pas prédéfinis par la CEI 62714. Le format de données CAEX central de langage AML prévoit des mécanismes de modélisation des données définies par l'utilisateur.

Afin de permettre l'échange des données définies par l'utilisateur, des accords et des fonctionnalités spécifiques à l'utilisateur peuvent par conséquent être nécessaires, mais ne font pas partie intégrante de la CEI 62714. Les méta-informations sources spécifiques aux outils techniques, décrites en 5.4, prennent en charge ces fonctionnalités.

Le langage AML permet de définir une relation entre les données définies par l'utilisateur et les données standard par l'intermédiaire des Role Concept, de PropertySet Concept ou des mappings CAEX standard. Ces concepts facilitent l'interprétation automatique des classes et attributs définis par l'utilisateur.

7.2 Attributs définis par l'utilisateur

Tous les attributs définis dans la CEI 62714 sont appelés attributs AML. Tous les attributs qui ne sont pas définis dans la CEI 62714 sont appelés attributs définis par l'utilisateur. Les attributs AML et les attributs définis par l'utilisateur sont archivés de la même manière que les attributs CAEX.

Les dispositions suivantes s'appliquent concernant les attributs définis par l'utilisateur:

- Les attributs CAEX doivent être archivés en langage AML conformément à la définition des attributs CAEX donnée en A.2.4 de la CEI 62424:2008.
- Si des unités sont requises, les attributs définis par l'utilisateur doivent reposer sur le même système d'unités. La présente partie de la CEI 62714 ne définit pas de système d'unités.

  Il est suggéré d'utiliser des unités SI conformément à l'ISO 80000-1. Pour les unités relatives à la technologie de l'information, il est suggéré d'utiliser la CEI 60027.

La Figure 17 donne un exemple d'objet “Object01” défini par l'utilisateur avec un attribut "Length" également défini par l'utilisateur.

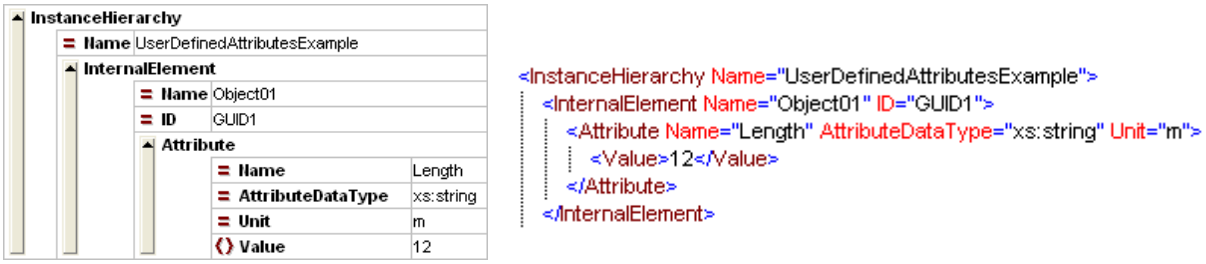


Figure 17 – Exemple d'attribut défini par l'utilisateur

7.3 InterfaceClasses définies par l'utilisateur

Toutes les InterfaceClasses définies dans la CEI 62714 sont appelées InterfaceClasses AML. Toutes les InterfaceClasses non définies dans la CEI 62714 sont appelées InterfaceClasses définies par l'utilisateur.

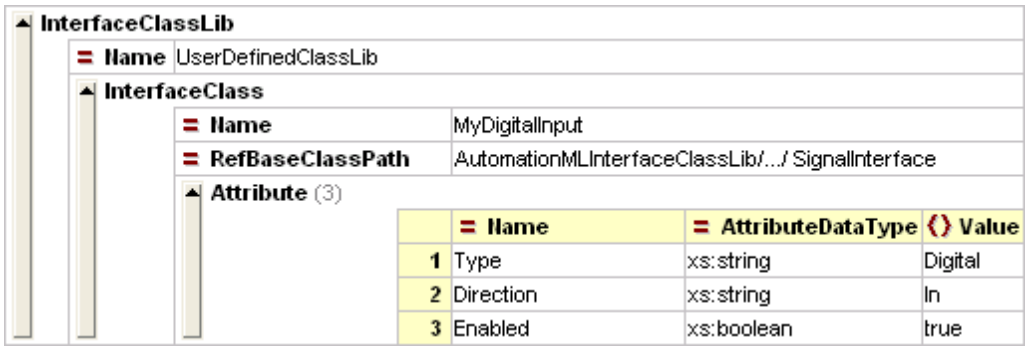
Les dispositions suivantes s'appliquent concernant les InterfaceClasses définies par l'utilisateur:

- Toutes les InterfaceClasses définies par l'utilisateur doivent être archivées conformément à la définition de l'InterfaceClass CAEX donnée en A.2.5 de la CEI 62424:2008.

  NOTE Les InterfaceClasses AML et les InterfaceClasses définies par l'utilisateur sont archivées de la même manière que les InterfaceClasses CAEX.
- Afin d'assurer l'interprétabilité algorithmique de la sémantique des InterfaceClasses définies par l'utilisateur, lesdites classes doivent être déduites des InterfaceClasses AML.

La Figure 18 montre un exemple de classe “MyDigitalInput” définie par l'utilisateur qui est déduite de l'InterfaceClass AML “SignalInterface”. Les relations d'héritage entre l'InterfaceClass “MyDigitalInput” et l'InterfaceClass AML standard “SignalInterface” permettent l'identification automatique de la classe définie par l'utilisateur comme interface à entrée numérique. Les attributs définis par l'utilisateur sont définis correctement. Dans cet exemple, les attributs définis par l'utilisateur ne font pas partie du domaine d'application de la présente partie de la CEI 62714.

NOTE Cet exemple utilise une notation réduite du chemin dédié à une lisibilité renforcée. Dans des applications réelles, le chemin est complet.




```
<InterfaceClassLib Name="UserDefinedClassLib">
  <InterfaceClass Name="MyDigitalInput" RefBaseClassPath="AutomationMLInterfaceClassLib/.../ SignalInterface">
    <Attribute Name="Type" AttributeDataType="xs:string">
      <Value>Digital</Value>
    </Attribute>
    <Attribute Name="Direction" AttributeDataType="xs:string">
      <Value>In</Value>
    </Attribute>
    <Attribute Name="Enabled" AttributeDataType="xs:boolean">
      <Value>true</Value>
    </Attribute>
  </InterfaceClass>
</InterfaceClassLib>
```

Figure 18 – Exemple d'InterfaceClass définie par l'utilisateur dans une bibliothèque InterfaceClassLib définie par l'utilisateur

7.4 RoleClasses définies par l'utilisateur

Toutes les RoleClasses définies dans la CEI 62714 sont appelées RoleClasses AML. Toutes les RoleClasses non définies dans la CEI 62714 sont appelées RoleClasses définies par l'utilisateur.

Les dispositions suivantes s'appliquent concernant les RoleClasses définies par l'utilisateur:

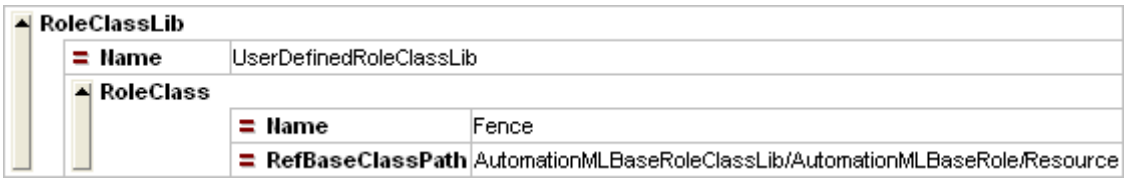
- Toutes les RoleClasses CAEX doivent être archivées conformément à la définition de la RoleClass CAEX donnée en A.2.6 de la CEI 62424:2008.

  NOTE 1 Les RoleClasses AML et les RoleClasses définies par l'utilisateur sont archivées de la même manière que les RoleClasses CAEX.

- Afin d'assurer l'interprétabilité sémantique des RoleClasses définies par l'utilisateur, lesdites classes doivent être déduites des RoleClasses AML.

  NOTE 2 Ceci permet l'interprétabilité algorithmique de la sémantique de la classe.

La Figure 19 montre un exemple de classe "Fence" définie par l'utilisateur qui est déduite de la RoleClass AML "Resource" standard. La relation d'héritage entre les classes "Fence" et "Resource" permet d'interpréter cette classe définie par l'utilisateur comme une ressource.




```
<RoleClassLib Name="UserDefinedRoleClassLib">
  <RoleClass Name="Fence" RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource"/>
</RoleClassLib>
```

Figure 19 – Exemple de RoleClass définie par l'utilisateur dans une bibliothèque RoleClassLib définie par l'utilisateur

7.5 SystemUnitClasses définies par l'utilisateur

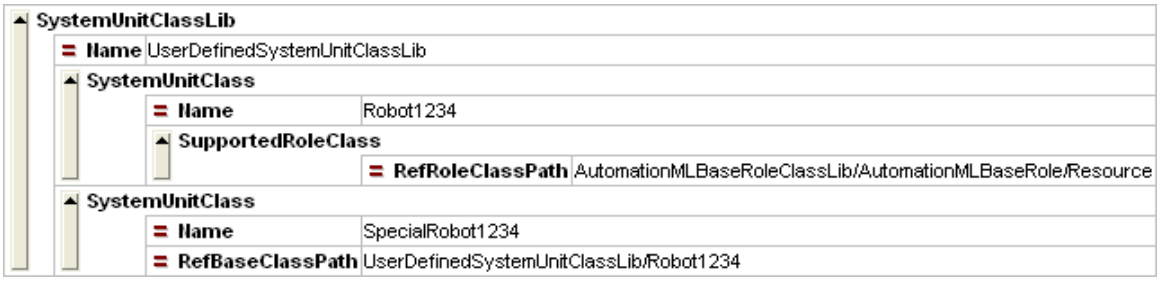
Toutes les SystemUnitClasses sont définies par l'utilisateur. La CEI 62714 ne spécifie pas de SystemUnitClasses.

Les dispositions suivantes s'appliquent concernant les SystemUnitClasses définies par l'utilisateur:

- Les SystemUnitClasses définies par l'utilisateur doivent être archivées en langage AML conformément à la définition des SystemUnitClasses CAEX donnée en A.2.3 de la CEI 62424:2008.
- Les SystemUnitClasses définies par l'utilisateur doivent être attribuées directement ou indirectement à une RoleClass AML et doivent utiliser les attributs AML dans toute la mesure du possible.

Exemples: La Figure 20 illustre la définition d'une SystemUnitClass définie par l'utilisateur, au moyen de deux exemples différents.

- La SystemUnitClass “Robot1234” illustre une classe définie par l'utilisateur qui prend en charge le rôle "Resource" de la bibliothèque RoleClassLib standard AML. Cette classe peut par conséquent être interprétée automatiquement comme une "Resource".
- La SystemUnitClass “SpecialRobot1234” illustre une nouvelle classe définie par l'utilisateur qui est déduite de la classe “Robot1234”. Cette classe est par conséquent également une ressource.




```
<SystemUnitClassLib Name="UserDefinedSystemUnitClassLib">
  <SystemUnitClass Name="Robot1234">
    <SupportedRoleClass RefRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource"/>
  </SystemUnitClass>
  <SystemUnitClass Name="SpecialRobot1234" RefBaseClassPath="UserDefinedSystemUnitClassLib/Robot1234"/>
</SystemUnitClassLib>
```

Figure 20 – Exemples de différentes SystemUnitClasses définies par l'utilisateur

7.6 InstanceHierarchies définies par l'utilisateur

Les InstanceHierarchies CAEX permettent d'archiver les informations techniques individuelles et celles liées à un projet. Elles constituent le cœur du format central AML et contiennent tous les objets de données individuels, y compris les propriétés, les interfaces, les relations et les références.

Les dispositions suivantes s'appliquent concernant les InstanceHierarchies définies par l'utilisateur:

- La présente partie de la CEI 62714 ne limite pas l'intensité des niveaux de hiérarchie.
- La présente partie de la CEI 62714 ne limite pas les règles d'architecture d'une hiérarchie.
- La présente partie de la CEI 62714 ne définit pas de conventions d'appellation pour les hiérarchies.
- Chaque objet AML d'une InstanceHierarchy doit être attribué directement ou indirectement à une RoleClasse AML afin de spécifier son type abstrait.

La Figure 21 illustre un exemple de hiérarchie de projet qui comprend une ligne "L001" avec un poste "S001" contenant deux robots “R0010_D” et “R0020_D”, ainsi qu'un transporteur “RF010” et un PLC “P001”.

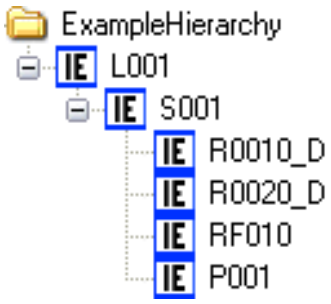


Figure 21 – Exemple d'InstanceHierarchy définie par l'utilisateur

La Figure 22 montre la représentation AML de cette structure. Conformément à la CEI 62424:2008, A.2.9, chaque objet est associé à une RoleClass.

```
<InstanceHierarchy Name="ExampleHierarchy">
  <InternalElement Name="L001" ID="GUID1">
    <InternalElement Name="S001" ID="GUID2">
      <InternalElement Name="R0010_D" ID="GUID3" RefBaseSystemUnitPath="RobotLibrary/Robot_1234">
        <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource"/>
      </InternalElement>
      <InternalElement Name="R0020_D" ID="GUID4" RefBaseSystemUnitPath="RobotLibrary/Robot_1234">
        <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource"/>
      </InternalElement>
      <InternalElement Name="RF010" ID="GUID5">
        <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource"/>
      </InternalElement>
      <InternalElement Name="P001" ID="GUID6">
        <RoleRequirements RefBaseRoleClassPath="AutomationMLCSRoleClassLib/ControlEquipment/ControlHardware/Controller/PLC"/>
      </InternalElement>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource/Structure"/>
    </InternalElement>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource/Structure"/>
  </InternalElement>
</InstanceHierarchy>
```

Figure 22 – Représentation AML d'une InstanceHierarchy définie par l'utilisateur

8 Concepts AML étendus

8.1 Vue d'ensemble générale

La présente partie de la CEI 62714 définit des concepts étendus pour la modélisation des aspects techniques spécifiques. Une vue d'ensemble informative et des exemples sont fournis en A.2.

8.2 Objet Port AML

Un port AML est un objet de même nature qui regroupe de nombreuses interfaces. Une vue d'ensemble informative du concept Port, y compris les exemples, est fournie en A.2.2.

Les dispositions suivantes s'appliquent concernant les ports AML:

- Un port AML doit être décrit par un InternalElement CAEX avec une association à la RoleClass "Port" décrite en 6.4.5.
- Un objet Port AML doit être modélisé en tant qu'objet enfant de l'objet ou de la classe AML considéré(e).
- Le regroupement d'interfaces requis doit être décrit par les ExternalInterfaces CAEX de l'objet Port.
- Un objet Port ne doit pas contenir d'InternalElements CAEX enfants.
- Il convient que toutes les ExternalInterfaces CAEX de l'objet Port soient déduites directement ou indirectement d'une classe d'interfaces AML définie en 6.3.
- Un port AML doit de plus comporter au moins une ExternalInterface CAEX déduite de l'InterfaceClass AML "PortConnector" décrite en 6.3.4.

NOTE Des dispositions normatives supplémentaires concernant les attributs de l'objet Port sont fournies en 6.4.5.

8.3 Objet Facet AML

Une Facette est un objet AML qui offre une vision secondaire des attributs ou des interfaces de l'objet AML parent. Ce concept permet l'archivage de différents paramètres de configuration tels que les données liées aux éléments IHM ou PLC, et permet par ailleurs l'automatisation de plusieurs étapes d'ingénierie de commande. Pour ce faire, la CEI 62714-1 définit la RoleClass AML "Facet" (voir 6.4.4). Une vue d'ensemble informative du concept Facet, y compris les exemples, est fournie en A.2.3.

Les dispositions suivantes s'appliquent concernant les objets Facet AML:

- Un objet Facet AML doit être décrit par un InternalElement CAEX avec une association à la RoleClass "Facet" décrite en 6.4.4.
- Un objet Facet AML doit être modélisé en tant qu'objet enfant de l'objet ou de la classe AML considéré(e).
- Les objets Facet doivent avoir un nom arbitraire unique parmi les objets jumeaux.

  NOTE Le nom d'un objet Facet est important pour l'association avec le concept Group. Voir A.2.3 pour la description et des exemples de concept.
- Un objet ou une classe AML peut avoir un nombre arbitraire d'objets Facet.
- Les objets Facet peuvent avoir un nombre arbitraire d'attributs de même nature.
- Un attribut Facet doit être associé à un attribut existant de l'objet AML parent, l'identifiant correspond pour sa part au même nom. Les attributs Facet qui ne font pas partie intégrante de l'objet parent ne sont pas admis.
- Une interface Facet doit être associée à une interface existante de l'objet parent, l'identifiant correspond pour sa part au même nom. Les interfaces Facet qui ne font pas partie intégrante de l'objet parent ne sont pas admises.

- Les objets Facet ne doivent pas contenir de nouveaux objets, attributs ou interfaces enfants.
- Ils ne doivent pas être imbriqués.
- Les objets Facet ne doivent pas modifier les attributs ou les interfaces existants.

8.4 Objet Group AML

Le concept "Group" AML permet de séparer les informations de structure des informations d'instance. Une vue d'ensemble informative du concept Group, y compris les exemples, est fournie en A.2.4.

Les dispositions suivantes s'appliquent concernant les objets "Group" AML:

- Un objet Group AML doit être décrit par un InternalElement CAEX avec une association à la RoleClass "Group" définie en 6.4.3.
- Un objet Group AML peut être modélisé à une position arbitraire de l'InstanceHierarchy ou d'une SystemUnitClass.
- Le nombre d'objets Group AML n'est pas limité.
- Un objet Group AML doit contenir uniquement les objets miroirs et/ou d'autres objets "Group".

  NOTE 1 Ainsi, les objets "Group" peuvent être imbriqués.

  NOTE 2 Si une instance A fait référence à une autre instance A*, A est appelée "objet miroir" et A* est appelée "objet maître" (conformément à la CEI 62424:2008, A.2.14). Un objet miroir référence l'objet maître et toutes les données de ce dernier. Ainsi, un objet miroir agit comme un pointeur de l'objet maître.
- Les objets Group AML ne doivent pas être utilisés pour décrire les hiérarchies d'une installation.
- Un objet Group AML peut archiver des informations supplémentaires en qualité d'attributs, d'interfaces ou de ports, afin de décrire les informations spécifiques aux groupes.

  NOTE 3 Ces attributs, ports et interfaces supplémentaires ne sont pas identiques aux attributs, ports ou interfaces des objets miroirs contenus.
- Il n'est pas admis de modifier les attributs, interfaces ou ports existants des objets miroirs ou d'ajouter des informations supplémentaires aux objets miroirs.
- Un objet miroir doit avoir un ID unique propre.

  NOTE 4 Un objet miroir est considéré identique à l'objet maître. L'ID prend en charge la différenciation entre la représentation miroir et l'objet maître.
- Si un objet maître est supprimé, tous les objets miroirs correspondants doivent être supprimés également afin d'éviter toutes incohérences.

  NOTE 5 Il s'agit d'une fonctionnalité outil qui ne relève pas du domaine d'application de la présente partie de la CEI 62714.
- La suppression éventuelle d'un objet miroir ne doit pas affecter l'objet maître.
- Lorsqu'il est utilisé, l'attribut “AssociatedFacet” doit avoir une valeur qui fournit un nom valide d'un objet Facet existant.

8.5 PropertySet AML

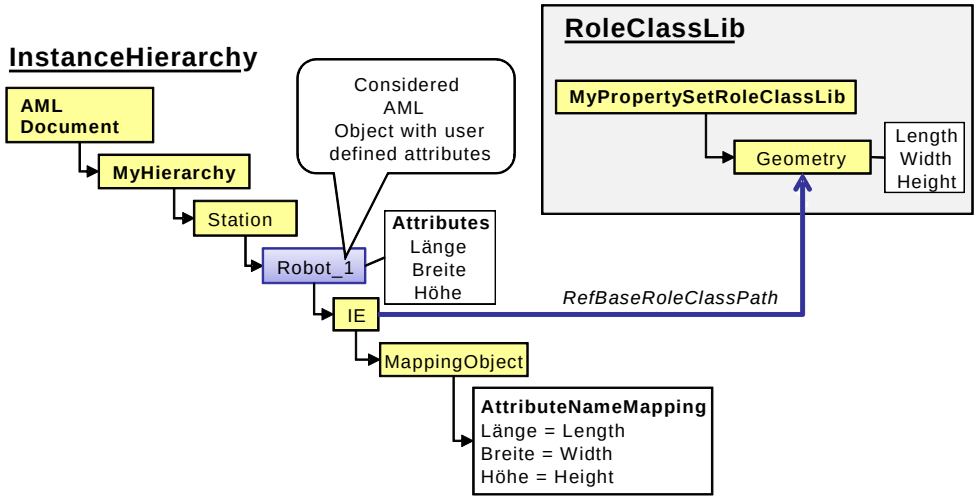
Un élément PropertySet est une classe de rôle contenant un ensemble d'attributs avec une syntaxe et une sémantique bien définies. Il est modélisé comme une classe de rôle déduite de la classe de rôle “PropertySet” standard. Le paragraphe A.2.5 fournit une vue d'ensemble conceptuelle.

Les dispositions suivantes s'appliquent concernant le concept PropertySet:

- Une classe PropertySet doit être modélisée en tant que classe de rôle et doit être déduite directement ou indirectement de la classe de rôle "PropertySet" standard.

- Les classes PropertySet peuvent être regroupées dans une ou plusieurs bibliothèques de classes de rôle.
- Les objets AML peuvent être associés à une ou plusieurs classes "Ensemble de propriétés".
- Pour chaque classe PropertySet d'un objet AML, un InternalElement CAEX enfant distinct dudit objet doit être créé, qui ne doit définir aucun attribut, aucune interface ou aucun InternalElement CAEX, sauf un nom et un ID. Cet objet enfant doit associer la classe de rôle PropertySet prévue au moyen de l'élément CAEX “RoleRequirement”.
- Les mappings entre les attributs de l'objet AML et un rôle PropertySet doivent être modélisés au moyen des éléments CAEX “MappingObject” et “AttributeNameMapping” de l'InternalElement enfant correspondant. Ces mappings sont valides entre l'objet AML d'appartenance et le rôle PropertySet référencé. Les attributs mis en correspondance doivent être reproduits dans la section RoleRequirement. Les attributs non mis en correspondance peuvent être reproduits dans la section RoleRequirement.
- Les attributs d'un rôle PropertySet peuvent être imbriqués.
- Les associations entre un objet AML et plusieurs ensembles de propriétés doivent être modélisées au moyen de plusieurs éléments enfants de l'objet AML, chaque élément ayant sa propre association RoleRequirement avec l'ensemble de propriétés correspondant, et ses propres mappings.

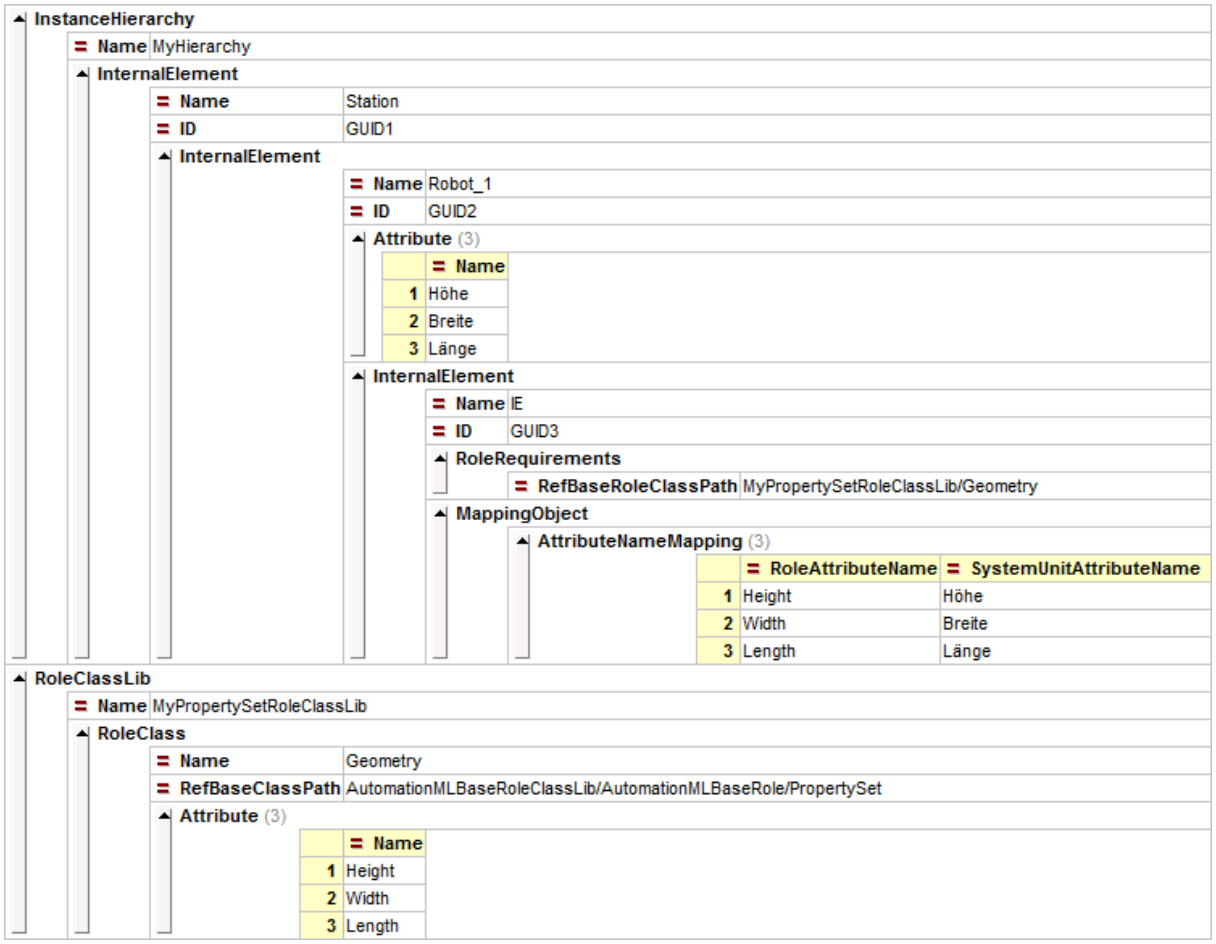
La Figure 23 illustre cette situation par un exemple. L'objet Robot_1 comporte un certain nombre d'attributs définis par l'utilisateur. Un IE InternalElement enfant est associé à l'objet PropertySet “Geometry” qui définit les attributs. Le MappingObject de IE spécifie le mapping des attributs propriétaires et des attributs normalisés.



Légende

Anglais	Français
AML document	Document AML
Considered AML Object with user defined attributes	Objet AML considéré avec attributs définis par l'utilisateur
Attributes	Attributs
Station	Poste
Length	Longueur
Width	Largeur
Height	Hauteur

Figure 23 – Exemple illustratif du concept PropertySet




```
<InstanceHierarchy Name="MyHierarchy">
   <InternalElement Name="Station" ID="GUID1">
      <InternalElement Name="Robot_1" ID="GUID2">
         <Attribute Name="Höhe"/>
         <Attribute Name="Breite"/>
         <Attribute Name="Länge"/>
         <InternalElement Name="IE" ID="GUID3">
            <RoleRequirements RefBaseRoleClassPath="MyPropertySetRoleClassLib/Geometry"/>
            <MappingObject>
               <AttributeNameMapping RoleAttributeName="Height" SystemUnitAttributeName="Höhe"/>
               <AttributeNameMapping RoleAttributeName="Width" SystemUnitAttributeName="Breite"/>
               <AttributeNameMapping RoleAttributeName="Length" SystemUnitAttributeName="Länge"/>
            </MappingObject>
         </InternalElement>
      </InternalElement>
   </InternalElement>
</InstanceHierarchy>
<RoleClassLib Name="MyPropertySetRoleClassLib">
   <RoleClass Name="Geometry" RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/PropertySet">
      <Attribute Name="Height"/>
      <Attribute Name="Width"/>
      <Attribute Name="Length"/>
   </RoleClass>
</RoleClassLib>
```

Figure 24 – Texte XML de l'exemple PropertySet

8.6 Prise en charge des rôles multiples

Outre le A.3.18 de la CEI 62424:2008, la présente partie de la CEI 62714 définit comment spécifier la prise en charge des rôles multiples pour une instance d'objet. Les rôles multiples sont intéressants si un objet peut avoir des fonctionnalités multiples. Un exemple qui illustre ces fonctionnalités multiples est un dispositif ayant la triple fonction conjointe de scanner, imprimante et télécopie. Le paragraphe A.2.7 donne une vue d'ensemble de ce dispositif et décrit un exemple correspondant.

Les dispositions suivantes s'appliquent concernant la prise en charge des rôles multiples:

- Si une instance prend en charge un seul rôle, le rôle correspondant doit être spécifié en utilisant l'attribut CAEX "RefBaseRoleClassPath" de l'élément RoleRequirement d'appartenance.

  NOTE 1 Cela est conforme au A.3.18 de la CEI 62424:2008, qui définit uniquement la prise en charge d'un rôle simultanément.

- Lorsqu'une instance prend en charge des rôles multiples, ces derniers doivent être définis en utilisant pour chacun d'entre eux un élément CAEX "SupportedRoleClass" en lieu et place de l'attribut CAEX "RefBaseRoleClassPath".

  NOTE 2 L'attribut "RefBaseRoleClassPath" peut être attribué uniquement une seule fois à l'élément RoleRequirement, tandis que l'élément CAEX "SupportedRoleClass" peut être défini plusieurs fois. Il s'agit de la condition fondamentale pour l'attribution de rôles multiples. Il s'agit cependant d'une faible extension sémantique conformément à la CEI 62424, qui ne modifie pas le format de données CAEX.

- Si une instance prend en charge des rôles multiples et si les exigences concernant les différents rôles doivent être archivées dans cette instance, cette opération doit être effectuée en utilisant l'élément CAEX "RoleRequirements", tandis que les attributs ou interfaces correspondants sont attribués directement, y compris le nom de rôle, une chaîne de séparation "." et le nom de l'attribut ou de l'interface.

  NOTE 3 Il s'agit d'une faible extension sémantique conformément à la CEI 62424, qui ne modifie pas le format de données CAEX. La différence avec A.3.18 de la CEI 62424:2008 réside dans le fait que le nom de rôle est ajouté à la définition de l'attribut ou de l'interface. Un exemple est décrit en A.2.7.

- Lorsque plusieurs classes de rôles prises en charge sont spécifiées et lorsque l'élément CAEX "RoleRequirements" associe conjointement un certain élément "RefBaseRoleClassPath", la classe de rôle associée constitue le rôle préférentiel. Dans ce cas, les définitions de l'attribut RoleRequirements et les mappings des attributs ou interfaces sans préfixe de nom de rôle explicite sont associés au rôle préférentiel.

  NOTE 4 Pour ce rôle préférentiel, l'utilisation est conforme à la CEI 62424:2008, Annexe A, sans extension sémantique.

8.7 Répartition des données centrales AML en différents documents

Conformément à la CEI 62424:2008, A.2.12, le format CAEX prend en charge de manière explicite la répartition des données techniques en différents fichiers et prévoit des mécanismes de référencement des fichiers CAEX externes au moyen de l'élément de même nature "ExternalReference" et du concept de pseudonyme correspondant du format CAEX.

8.8 Internationalisation

Différents langages, par exemple, des noms et des descriptions, peuvent être archivés en langage AML conformément aux spécifications XML basées sur le format UTF-8.

8.9 Informations de version des objets AML

Les champs standard de version et de révision conformes au A.2.2.2 de la CEI 62424:2008 doivent être utilisés pour l'archivage des informations de version et de révision des objets AML individuels (instances d'objets).

Pour l'archivage des informations de version liées au format AML et des informations de version liées aux bibliothèques AML, voir 5.3.

Pour l'archivage des méta-informations spécifiques aux outils, voir 5.4.

Annexe A
(informative)

Introduction générale au langage Automation Markup Language

A.1 Concepts généraux relatifs au langage Automation Markup Language

A.1.1 Architecture Automation Markup Language

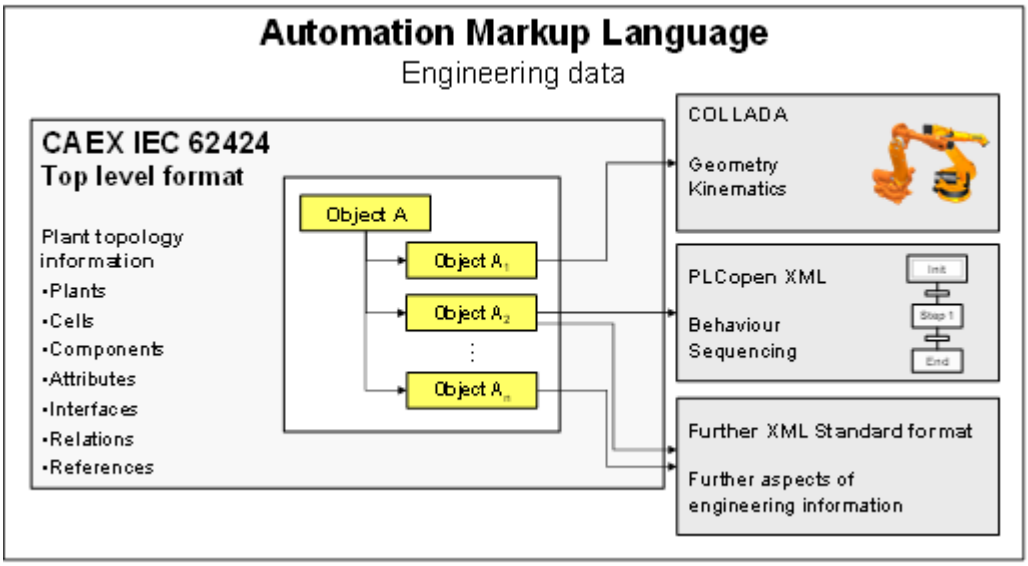
Automation Markup Language (Langage de balisage d'automatisation) est un format de données de type schéma XML conçu pour l'échange des informations concernant l'ingénierie d'usine, indépendamment du fournisseur. L'objectif du langage AML est l'interconnexion des outils techniques, issus de l'environnement d'outils hétérogène existant, dans leurs différentes disciplines, par exemple, ingénierie des installations mécaniques, études d'électricité, ingénierie de procédés, ingénierie de commande de processus, développement des IHM, programmation PLC, programmation de robots, etc.

Le langage AML archive les informations techniques en respectant le paradigme orienté objet, et permet la modélisation des composants d'installations réelles sous forme d'objets de données qui englobent différents aspects. Un objet peut comporter d'autres sous-objets, et peut lui-même faire partie intégrante d'une composition ou d'une agrégation plus importante. Il peut décrire par exemple un signal, un PLC, un réservoir, une vanne de régulation, un robot, une cellule de fabrication avec différents niveaux de détail ou un site, une chaîne ou une installation dans leur intégralité. Les objets typiques que l'on trouve dans l'automatisation d'installations comprennent les informations concernant la topologie, la géométrie, la cinématique et la logique, tandis que la logique comprend pour sa part le séquencement, le comportement et la commande.

AML combine les formats de données industrielles existants, conçus pour l'archivage et l'échange de différents aspects des informations techniques. Ces formats de données sont utilisés “en l'état” dans le cadre de leurs propres spécifications et ne sont pas associés aux besoins du langage AML.

La caractéristique centrale de l’AML est le format de données central CAEX qui connecte les différents formats de données. Le langage AML a par conséquent une architecture de document répartie intrinsèque.

La Figure A.1 illustre l'architecture AML de base et la répartition des informations concernant la topologie, la géométrie, la cinématique et la logique.



Légende

Anglais	Français
Engineering data	Données techniques
CAEX IEC 62424 top level format	Format central défini dans la CEI 62424, de type CAEX
Object	Objet
Plant topology information	Informations concernant la topologie de l'installation
Plants	Installations
Cells	Cellules
Components	Composants
Attributes	Attributs
References	Références
Geometry	Géométrie
Kinematics	Cinématique
Behaviour	Comportement
Sequencing	Séquencement
Further XML Standard format	Autre format standard XML
Further aspects of engineering information	Autres aspects des informations techniques

Figure A.1 – Architecture générale AML

Les principaux avantages du concept de documents répartis sont l'utilisation de formats de données éprouvés et établis, la répartition des données dans des fichiers différents, ce qui facilite le traitement des informations volumineuses et l'utilisation simplifiée des fichiers de bibliothèque AML susceptibles d'être archivés, échangés et consultés séparément. Enfin, différents niveaux de détail, par exemple, des variantes géométriques, peuvent faire l'objet d'un archivage séparément. AML définit principalement les associations entre les formats de données référencés et les objets de technologie.

Le paragraphe A.1.1, à l'aide d'exemples succincts, présente de manière générale les informations susceptibles d'être archivées et échangées au moyen du langage AML.

- **Informations concernant la topologie de l'installation**: La topologie de l'installation décrit une installation comme une structure hiérarchique des objets individuels constitutifs de l'installation, qui sont représentés par des objets de données individuels. La

modélisation de la structure d'un objet atteint un certain niveau de détail (par exemple, robot, outil de préhension, mais non des essieux ou des assemblages/joints); les objets comprennent les propriétés et les relations avec d'autres objets dans leur structure hiérarchique. La topologie de l'installation agit comme la structure de données centrale et est archivée au moyen du format de données CAEX conformément à la CEI 62424:2008, Article 7, Annexe A et Annexe C. Par extension à la CEI 62424:2008, AML définit les références entre les objets CAEX et les informations archivées dans les documents externes hors du format CAEX. Le paragraphe A.1.2 fournit des exemples de modélisation des informations concernant la topologie de l'installation au moyen du langage AML.

- **Informations concernant la géométrie et la cinématique**: La géométrie d'un objet d'installation unique comprend sa représentation géométrique. Les informations concernant la cinématique décrivent les connexions physiques des objets solides tridimensionnels, ainsi que les dépendances entre les objets. Les informations tant géométriques que cinématiques sont archivées au moyen du format de fichier COLLADA. De plus, le fichier COLLADA inclut la définition de l'association des informations concernant la géométrie et la cinématique. Les interfaces COLLADA peuvent être éditées en tant que ExternalInterfaces CAEX dans le format central destiné à une interconnexion ultérieure. En dehors des informations concernant la géométrie COLLADA de différents objets, une scène complète peut être déduite automatiquement. Ces fichiers peuvent être référencés à partir du format CAEX et peuvent être interconnectés au moyen des mécanismes de liaison CAEX. Le paragraphe A.1.3 fournit un exemple succinct. Les détails sont à spécifier dans la CEI 62714-3.
- **Informations concernant la logique**: Les informations concernant la logique décrivent des séquences d'actions et le comportement des objets, y compris les connexions E/S et les variables logiques. Les séquences sont décrites et archivées dans des documents XML PLCopen externes. Les variables ou les signaux peuvent être édités sous forme de ExternalInterfaces CAEX. Ces documents peuvent être référencés en un format autre que CAEX et peuvent être interconnectés avec le format CAEX. Le paragraphe A.1.4 fournit une brève introduction concernant les principaux concepts. Les détails sont à spécifier dans la CEI 62714-4.
- **Informations concernant les références et les relations**: AML différencie les références des relations. Les références illustrent les liaisons entre les objets CAEX et les informations à archivage externe. Les relations illustrent les associations entre les objets CAEX. De plus, le même mécanisme permet d'archiver les associations entre les informations archivées dans des documents externes. Pour ce faire, il est nécessaire d'éditer les partenaires de liaison associés au moyen des ExternalInterfaces CAEX dans la topologie de l'installation de même nature. Les détails concernant le référencement des documents COLLADA et XML PLCopen sont à spécifier dans la CEI 62714-3 et la CEI 62714-4. Le paragraphe A.1.5 fournit une vue d'ensemble informative concernant la modélisation des références et des relations avec le langage AML. Les paragraphes 5.6 et 5.7 spécifient les dispositions normatives.
- **Référencement d'autres formats de données**: L'extension future de la CEI 62714 est possible par la publication de parties supplémentaires spécifiant l'intégration d'autres formats de données qui utilisent les mécanismes de référence AML.

L'échange des informations techniques requiert par ailleurs certains concepts étendus. L'Article A.2 explicite ces concepts et l'Article 8 spécifie leurs dispositions normatives.

NOTE Dans le présent document, les chemins sont parfois illustrés sous une forme réduite, par exemple, "AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Port" au lieu de la notation "AutomationMLInterfaceClassLib/.../Port". Cela permet la lisibilité du document. Dans les documents XML réels, tous les chemins sont archivés conformément à la présente partie de la CEI 62714.

A.1.2 Modélisation des informations concernant la topologie de l'installation

Dans le langage AML, les composants d'installations réelles sont modélisés en tant qu'objets de données qui englobent différents aspects des informations techniques. Pour ce faire, il est nécessaire de structurer les objets de données. Une méthode établie de structuration de ce type d'objets de données est une hiérarchie des objets qui correspond à la topologie de l'installation (voir 3.1.20).

En vue de l'archivage des structures d'installation hiérarchiques, AML utilise des concepts fournis par le format de données central CAEX conformément à la CEI 62424:2008, A.2.11. La Figure A.2 présente un exemple de topologie de l'installation d'une chaîne de fabrication comportant plusieurs objets de niveaux hiérarchiques différents.

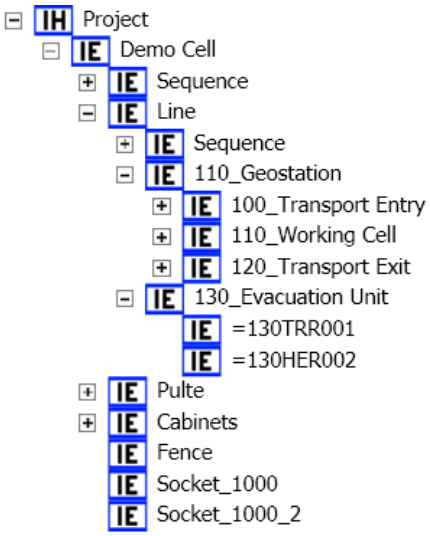


Figure A.2 – Topologie de l'installation avec AML

Les concepts CAEX standard permettent de modéliser les hiérarchies multiples, les structures croisées et les réseaux d'objets complexes. Cependant, l'élément clé de l'efficacité du paradigme orienté objet réside dans la disponibilité de bibliothèques contenant des solutions éprouvées prédéfinies. Pour ce faire, le format CAEX prévoit de nombreux types de bibliothèque différents pour les interfaces, les rôles, les unités centrales et les hiérarchies d'instances importants pour le langage AML.

- **InterfaceClasses et InterfaceClassLib:** Les interfaces permettent de définir les relations entre objets AML. Le paragraphe 6.3 spécifie la relation AML-InterfaceClassLib standard avec un ensemble d'InterfaceClasses abstraites dédiées aux systèmes d'automatisation généraux. Ces classes comportent une définition syntaxique et sémantique, et permettent également de spécifier des interfaces d'objets définies par l'utilisateur. Le paragraphe 7.3 spécifie la modélisation de classes de rôles définies par l'utilisateur.
- **RoleClasses et RoleClassLib:** Les RoleClasses permettent de définir les caractéristiques abstraites des objets CAEX et permettent ainsi l'interprétation sémantique automatique des objets AML définis par l'utilisateur. Le paragraphe 6.4 spécifie la relation AML-RoleClassLib de base avec un ensemble de RoleClasses abstraites dédiées aux systèmes d'automatisation généraux. Le paragraphe 7.4 spécifie la modélisation de classes de rôles définies par l'utilisateur. Il s'agit de spécifier d'autres bibliothèques de rôle dans la CEI 62714-2.
- **SystemUnits et SystemUnitClassLib:** La bibliothèque SystemUnitClassLib permet d'archiver les classes AML spécifiques au fournisseur. Le paragraphe 7.5 spécifie les règles d'architecture qui permettent de définir les SystemUnitClasses. AML ne prédéfinit pas une bibliothèque SystemUnitClassLib ou SystemUnitClass donnée.
- **Instances et InstanceHierarchy:** Les hiérarchies d'instances archivent les données de topologie des projets réels et constituent par conséquent le noyau du langage AML. Elles se composent d'instances d'objet AML. Le paragraphe 7.6 spécifie la méthode d'archivage des informations techniques au moyen des hiérarchies d'instances de type InstanceHierarchy.

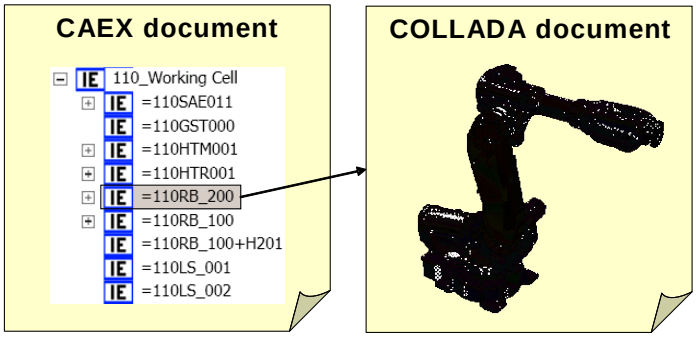
Un aspect important de la modélisation d'une topologie de l'installation est l'identification des objets. Différents outils techniques utilisent des concepts différents pour l'identification des objets, par exemple, un nom, un identifiant ou un chemin unique. Certains outils autorisent les changements des identifiants au cours de la durée de vie, et d'autres pas. Ce principe

s'applique parfaitement pour un outil donné, l'échange des objets entre différents outils n'étant toutefois pas possible. Pour ce faire, le 5.5 spécifie un concept d'identification d'objets obligatoire. Seul ce type de concept permet l'échange de données entre différents outils techniques avec des concepts d'identification d'objets individuels.

A.1.3 Référencement des informations concernant la géométrie et la cinématique

Les informations concernant la géométrie et la cinématique sont archivées dans des documents distincts suivant le format de données COLLADA. La modélisation des informations concernant la géométrie et la cinématique est par conséquent double. D'une part, l'objet correspondant est modélisé selon le format CAEX sans aucune information concernant la géométrie ou la cinématique, comme cela est décrit dans la présente partie de la CEI 62714. D'autre part, un document COLLADA est à fournir, contenant les informations concernant la géométrie et la cinématique. Enfin, l'objet CAEX archive une référence au document COLLADA comme cela est à décrire dans la CEI 62714-3.

La Figure A.3 présente un exemple de document AML comprenant l'objet "110RB_200", qui référence un document COLLADA externe contenant les informations correspondantes concernant la géométrie et la cinématique.



Légende

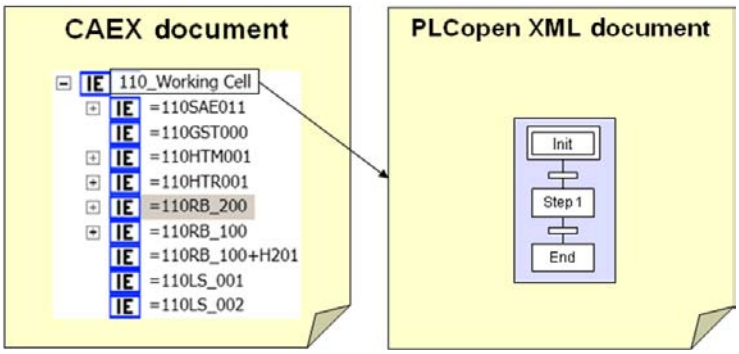
Anglais	Français
CAEX document	Document CAEX
COLLADA document	Document COLLADA

Figure A.3 – Référence entre le format CAEX et un document COLLADA

La référence est modélisée au moyen d'une interface CAEX déduite de la classe d'interface AML standard “COLLADAInterface” spécifiée en 5.7 et 6.3.7. Les détails sont à spécifier dans la CEI 62714-3.

A.1.4 Référencement des informations concernant la logique

Les informations concernant la logique sont archivées dans des documents distincts suivant le format de données XML PLCopen. La modélisation des informations concernant la logique est par conséquent double. D'une part, l'objet correspondant est modélisé selon le format CAEX sans aucune information concernant la logique, comme cela est décrit dans la présente partie de la CEI 62714. D'autre part, un document XML PLCopen est à fournir, contenant les informations concernant la logique, comme cela est à décrire dans la CEI 62714-4. Enfin, l'objet CAEX archive une référence au document XML PLCopen. La Figure A.4 présente un exemple de document AML comprenant l'objet "110_Working Cell“, qui référence un document XML PLCopen externe contenant les informations correspondantes concernant la logique.



Légende

Anglais	Français
CAEX document	Document CAEX
PLCopen document	Document PLCopen

Figure A.4 – Référence entre le format CAEX et un document XML PLCopen

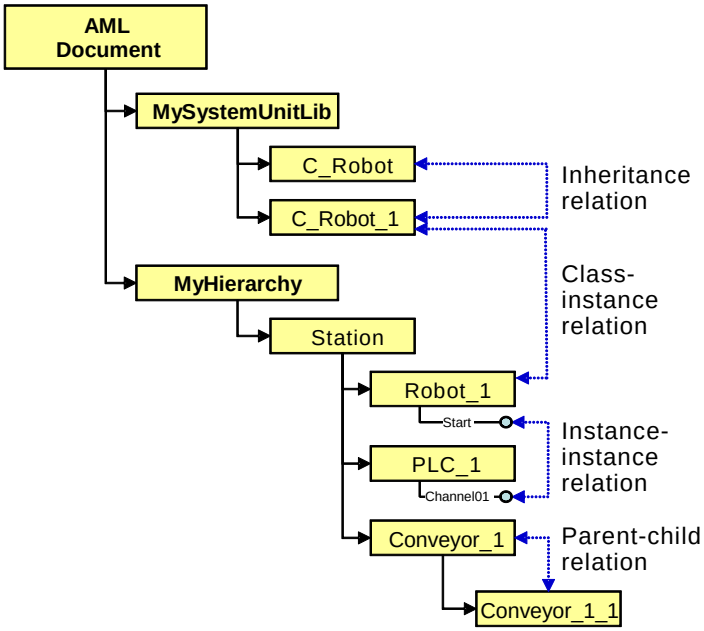
A.1.5 Modélisation des relations

La modélisation des objets rend nécessaire de définir des mécanismes qui permettent de déterminer les objets les uns par rapport aux autres. Des mécanismes supplémentaires sont nécessaires pour relier ces objets avec des données stockées en externe.

Une relation exprime une association entre deux objets ou plus. Cette dépendance peut être de toute nature, y compris les dépendances physiques et logiques. AML prend en charge les relations suivantes:

- Relations parent-enfant (voir 5.6.2 et 5.6.3)
  - relations parent-enfant entre les objets AML
  - relations parent-enfant entre les classes AML
- Relations d'héritage (voir 5.6.4)
  - relations d'héritage entre SystemUnitClasses
  - relations d'héritage entre RoleClasses
  - relations d'héritage entre InterfaceClasses
- Relations entre classe et instance (voir 5.6.5)
  - relation entre une SystemUnitClass et une de ses instances
  - relation entre une RoleClass et une de ses instances
  - relation entre une InterfaceClass et une de ses instances
- Relations entre instances (voir 5.6.6)
  - relations entre les objets AML
  - relations entre les données éditées et stockées en externe

La Figure A.5 présente les types de relation mentionnés pris en charge par AML à l'aide d'un exemple.



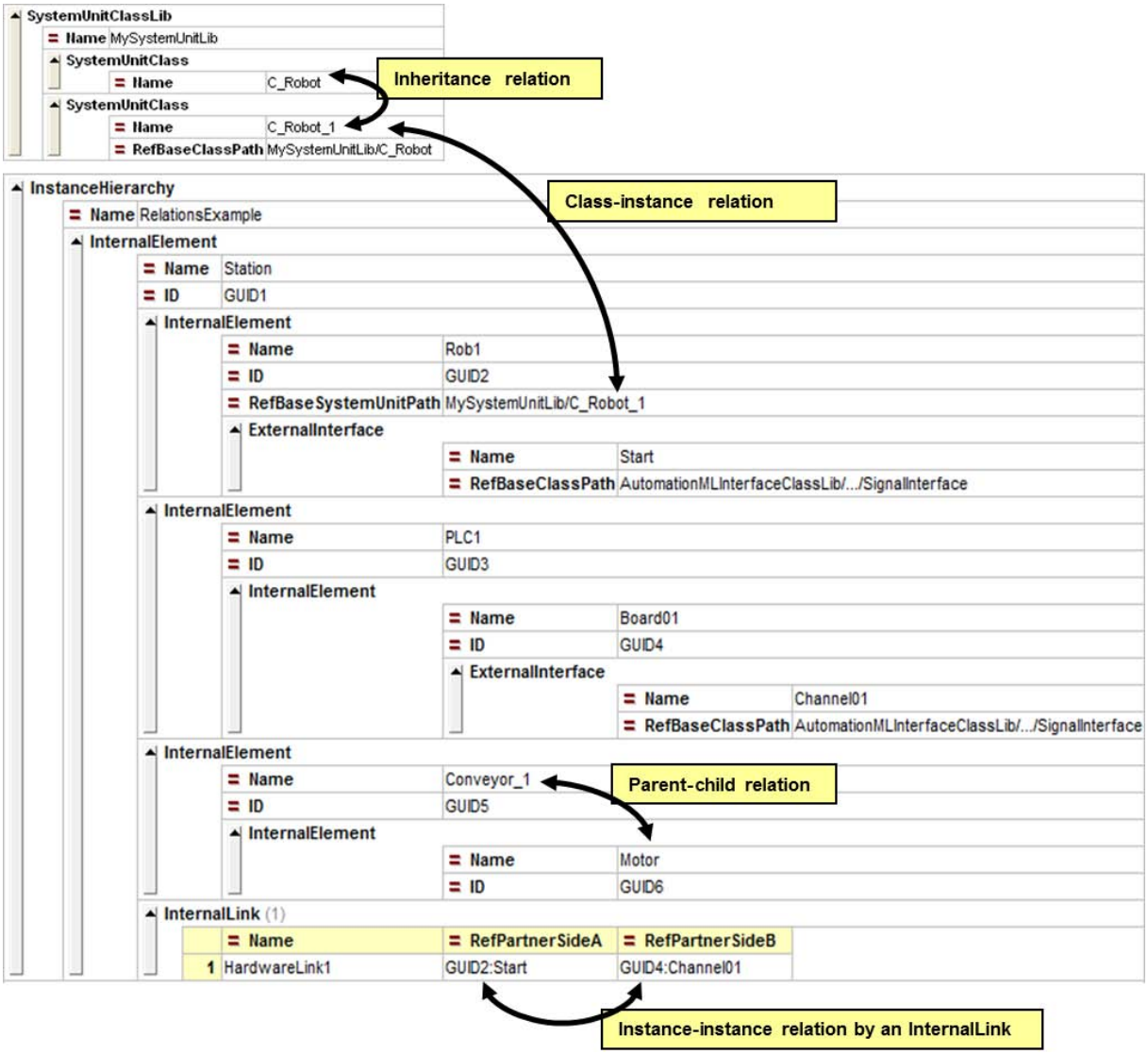
Légende

Anglais	Français
AML document	Document AML
Inheritance relation	Relation d'héritage
Class-instance relation	Relation classe-instance
Instance-instance relation	Relation entre instances
Parent-child-relation	Relation parent-enfant
Station	Poste
Start	Début

Figure A.5 – Relations dans le langage AML

La Figure A.6 illustre le modèle AML qui correspond à l'exemple au moyen d'une vue de tableau. Noter que le paramètre fictif "/…/", destiné à améliorer la lisibilité permet de réduire grandement les informations concernant le chemin. La Figure A.7 présente le texte XML correspondant de la bibliothèque AML.

La Figure A.8 présente le texte XML correspondant de la InstanceHierarchy.



Légende

Anglais	Français
Inheritance relation	Relation d'héritage
Class-instance-relation	Relation classe-instance
Parent-child relation	Relation parent-enfant
Instance-instance relation by an InternalLink	Relation entre instances par une InternalLink

Figure A.6 – Description XML de l'exemple illustratif des relations

```
<SystemUnitClassLib Name="MySystemUnitLib">
  <SystemUnitClass Name="C_Robot"/>
  <SystemUnitClass Name="C_Robot_1" RefBaseClassPath="MySystemUnitLib/C_Robot"/>
</SystemUnitClassLib>
```

Figure A.7 – Texte XML de la bibliothèque SystemUnitClassLib de l'exemple illustratif des relations

```
<InstanceHierarchy Name="RelationsExample">
  <InternalElement Name="Station" ID="GUID1">
    <InternalElement Name="Rob1" ID="GUID2" RefBaseSystemUnitPath="MySystemUnitLib/C_Robot_1">
      <ExternalInterface Name="Start" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface"/>
    </InternalElement>
    <InternalElement Name="PLC1" ID="GUID3">
      <InternalElement Name="Board01" ID="GUID4">
        <ExternalInterface Name="Channel01" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface"/>
      </InternalElement>
    </InternalElement>
    <InternalElement Name="Conveyor_1" ID="GUID5">
      <InternalElement Name="Motor" ID="GUID6"/>
    </InternalElement>
    <InternalLink Name="HardwareLink1" RefPartnerSideA="GUID2:Start" RefPartnerSideB="GUID4:Channel01"/>
  </InternalElement>
```

Figure A.8 – Texte XML de la InstanceHierarchy de l'exemple illustratif des relations

A.2 Concepts et exemples AML étendus

A.2.1 Vue d'ensemble générale

AML définit des concepts étendus pour la modélisation des aspects techniques spécifiques tels que le concept AML Port, le concept AML Facet et le concept AML Group. Le Tableau A.1 présente de manière succincte ces concepts.

Tableau A.1 – Vue d'ensemble des principaux concepts AML étendus

Concept	Description
AML Port	Le concept Port permet une description très détaillée des interfaces complexes. Les AML Ports sont constitués d'un ensemble d'interfaces AML associées. Ils peuvent être assimilés aux fiches ou aux prises femelles.
AML Facet	Les AML Facet permettent l'archivage d'un sous-ensemble d'attributs et d'interfaces d'un objet AML. Elles peuvent être considérées comme des éléments de vision des données techniques.
AML Group	Les AML Group permettent l'archivage d'éléments d'observation distincts d'un sous-ensemble d'objets AML. Ils peuvent être utilisés pour le filtrage des objets de l'arborescence de l'installation pour différents outils techniques.
PropertySet	Le concept PropertySet permet le mapping des attributs propriétaires des objets AML définis par l'utilisateur, avec des attributs prédéfinis de manière sémantique. Ces attributs à convention sémantique sont archivés dans des classes de rôles PropertySet.
Process-Product-Resource	Le concept Process-Product-Resource permet la structuration de haut niveau des données techniques, établie sur une perspective centrée sur le processus, le produit ou les ressources, y compris les relations entre ces données.

A.2.2 Concept AML Port

A.2.2.1 Description du concept

Un AML Port est un objet de même nature qui regroupe de nombreuses interfaces (voir Figure A.9). Un objet Port appartient à un objet AML parent et décrit les interfaces complexes

de l'objet parent. Les Ports peuvent être connectés entre eux à un niveau d'abstraction supérieur, en lieu et place d'une liaison de chaque interface simple. Les AML Ports sont utiles pour décrire les fiches, les prises femelles ou les autres groupes éventuels d'interfaces qui peuvent être directement connectés entre eux. Pour ce faire, AML définit la RoleClass AML "Port" (voir 6.4.5). Des dispositions normatives sont spécifiées en 8.2.

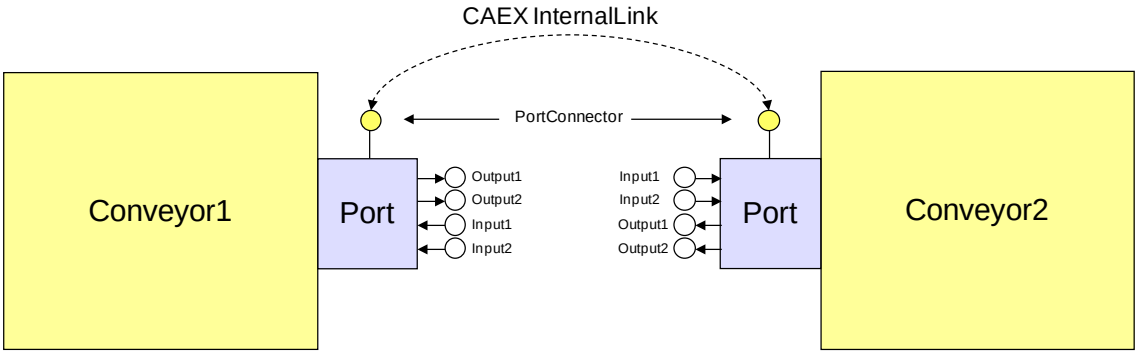
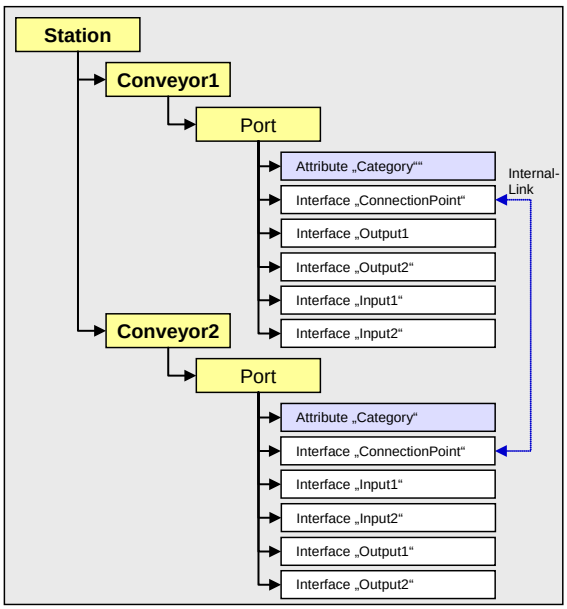


Figure A.9 – Concept Port

A.2.2.2 Exemple

La Figure A.10 donne un exemple de concept AML Port. L'objet "Station" comprend les sous-objets “Conveyor1” et “Conveyor2”. Ces deux sous-objets ont chacun un objet Port. L'objet Port comprend un ensemble d'interfaces, ainsi qu'une interface standard “ConnectionPoint” déduite de l'InterfaceClass AML "PortConnector". L'interface définie dans la présente norme peut être reliée à l'aide d'un InternalLink CAEX. Cette relation signifie que les deux ports sont connectés entre eux. La liaison interne des sous-interfaces n'est pas décrite de manière détaillée, et seuls les ConnectionPoints abstraits sont connectés. Outre ce concept, AML permet l'archivage de chaque liaison individuelle entre les sous-interfaces.

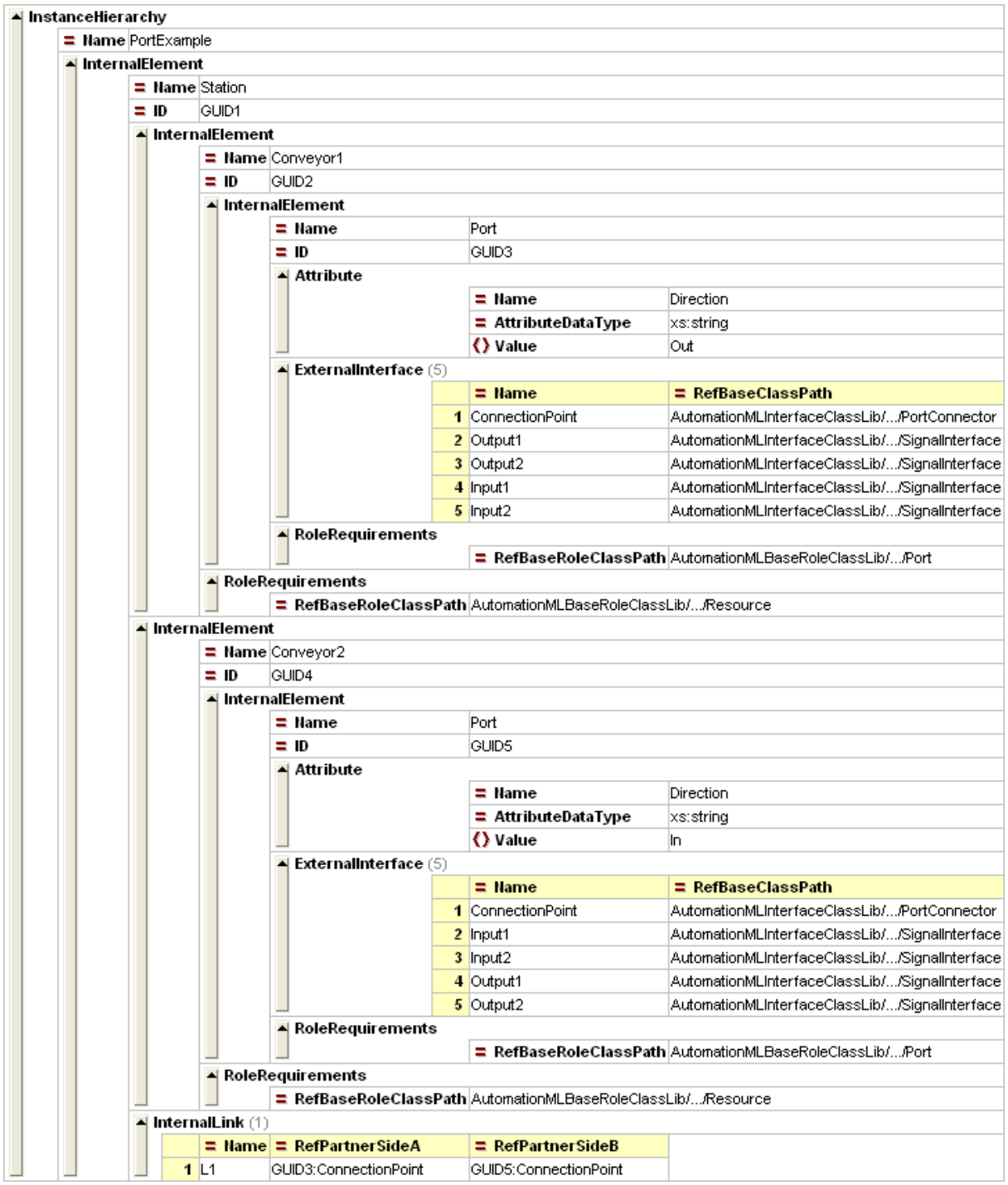


Légende

Anglais	Français
Station	Poste
Attribute	Attribut

Figure A.10 – Exemple de description du concept AML Port

La Figure A.11 et la Figure A.12 décrivent la mise en œuvre AML de l'exemple de système décrit à la Figure A.10.



InstanceHierarchy

Name	PortExample

InternalElement

Name	Station
ID	GUID1

InternalElement

Name	Conveyor1
ID	GUID2

InternalElement

Name	Port
ID	GUID3

Attribute

Name	Direction
AttributeDataType	xs:string
Value	Out

ExternalInterface (5)

	Name	RefBaseClassPath
1	ConnectionPoint	AutomationMLInterfaceClassLib/.../PortConnector
2	Output1	AutomationMLInterfaceClassLib/.../SignalInterface
3	Output2	AutomationMLInterfaceClassLib/.../SignalInterface
4	Input1	AutomationMLInterfaceClassLib/.../SignalInterface
5	Input2	AutomationMLInterfaceClassLib/.../SignalInterface

RoleRequirements

RefBaseRoleClassPath	AutomationMLBaseRoleClassLib/.../Port

RoleRequirements

RefBaseRoleClassPath	AutomationMLBaseRoleClassLib/.../Resource

InternalElement

Name	Conveyor2
ID	GUID4

InternalElement

Name	Port
ID	GUID5

Attribute

Name	Direction
AttributeDataType	xs:string
Value	In

ExternalInterface (5)

	Name	RefBaseClassPath
1	ConnectionPoint	AutomationMLInterfaceClassLib/.../PortConnector
2	Input1	AutomationMLInterfaceClassLib/.../SignalInterface
3	Input2	AutomationMLInterfaceClassLib/.../SignalInterface
4	Output1	AutomationMLInterfaceClassLib/.../SignalInterface
5	Output2	AutomationMLInterfaceClassLib/.../SignalInterface

RoleRequirements

RefBaseRoleClassPath	AutomationMLBaseRoleClassLib/.../Port

RoleRequirements

RefBaseRoleClassPath	AutomationMLBaseRoleClassLib/.../Resource

InternalLink (1)

	Name	RefPartnerSideA	RefPartnerSideB
1	L1	GUID3:ConnectionPoint	GUID5:ConnectionPoint

Figure A.11 – Description XML du concept AML Port

```
<InstanceHierarchy Name="PortExample">
  <InternalElement Name="Station" ID="GUID1">
    <InternalElement Name="Conveyor1" ID="GUID2">
      <InternalElement Name="Port" ID="GUID3">
        <Attribute Name="Direction" AttributeDataType="xs:string">
          <Value>Out</Value>
        </Attribute>
        <ExternalInterface Name="ConnectionPoint" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PortConnector"/>
        <ExternalInterface Name="Output1" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface"/>
        <ExternalInterface Name="Output2" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface"/>
        <ExternalInterface Name="Input1" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface"/>
        <ExternalInterface Name="Input2" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Port"/>
      </InternalElement>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
    </InternalElement>
    <InternalElement Name="Conveyor2" ID="GUID4">
      <InternalElement Name="Port" ID="GUID5">
        <Attribute Name="Direction" AttributeDataType="xs:string">
          <Value>In</Value>
        </Attribute>
        <ExternalInterface Name="ConnectionPoint" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PortConnector"/>
        <ExternalInterface Name="Input1" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface"/>
        <ExternalInterface Name="Input2" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface"/>
        <ExternalInterface Name="Output1" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface"/>
        <ExternalInterface Name="Output2" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Port"/>
      </InternalElement>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
    </InternalElement>
    <InternalLink Name="L1" RefPartnerSideA="GUID3:ConnectionPoint" RefPartnerSideB="GUID5:ConnectionPoint"/>
  </InternalElement>
</InstanceHierarchy>
```

Figure A.12 – Texte XML de description du concept AML Port

A.2.2.3 Modélisation d'un Port en tant que SystemUnitClass AML définie par l'utilisateur

L'exemple suivant de la Figure A.13 illustre une description XML d'une SystemUnitClass “myPortClass” définie par l'utilisateur.

SystemUnitClass		
= Name	myPortClass	
Attribute (2)		
	= Name	Value
1	Direction	InOut
2	Category	MaterialFlow
ExternalInterface		
	= Name	ConnectionPoint
	= RefBaseClassPath	AutomationMLInterfaceClassLib/.../PortConnector
SupportedRoleClass		
	= RefRoleClassPath	AutomationMLBaseRoleClassLib/.../Port

```
<SystemUnitClass Name="myPortClass">
  <Attribute Name="Direction">
    <Value>InOut</Value>
  </Attribute>
  <Attribute Name="Category">
    <Value>MaterialFlow</Value>
  </Attribute>
  <ExternalInterface Name="ConnectionPoint" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PortConnector"/>
  <SupportedRoleClass RefRoleClassPath="AutomationMLBaseRoleClassLib/.../Port"/>
</SystemUnitClass>
```

Figure A.13 – Définition d'une classe AML Port “myPortClass” définie par l'utilisateur

A.2.3 Concept AML Facet

A.2.3.1 Description du concept

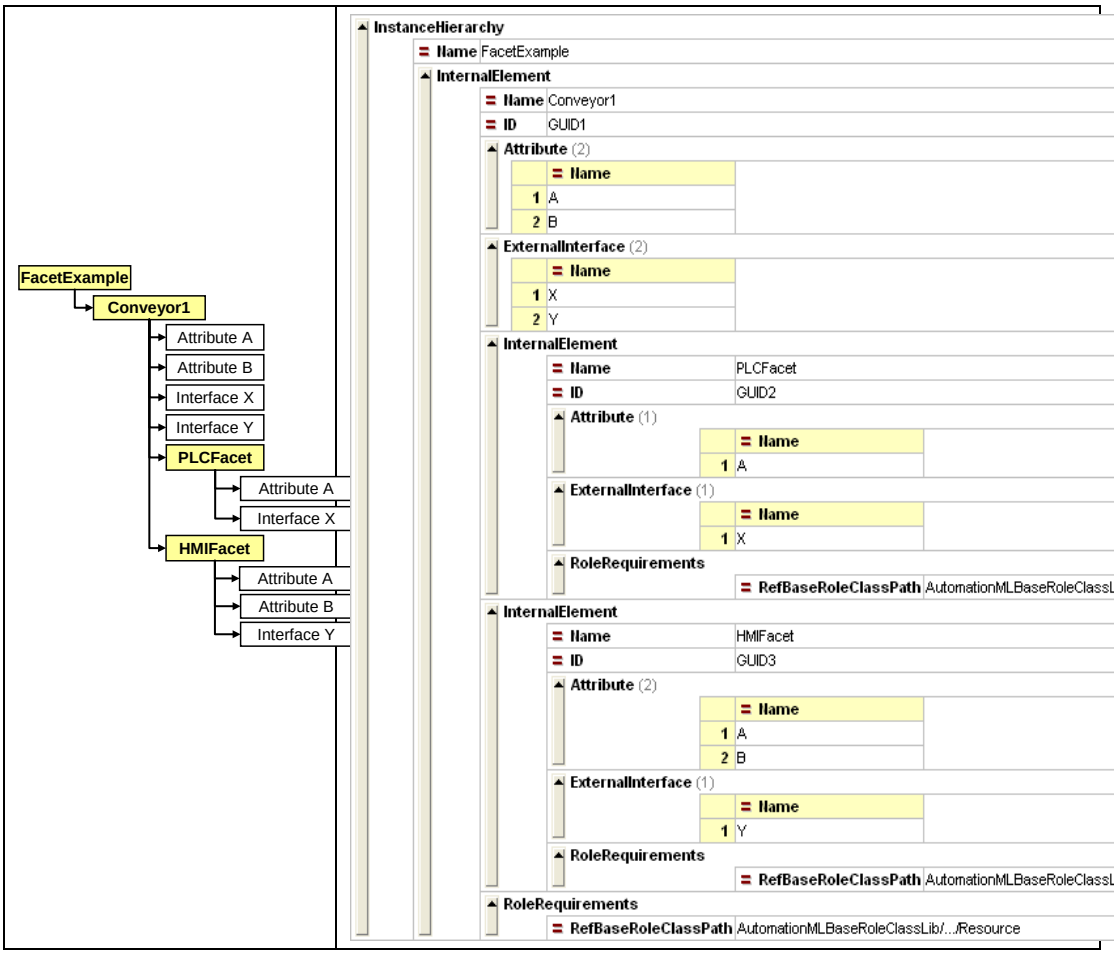
Un objet Facet est un objet AML qui offre une vision secondaire des attributs ou des interfaces de l'objet AML parent. Ce concept permet l'archivage de différents paramètres de configuration tels que les données liées aux éléments IHM ou PLC, et permet par ailleurs l'automatisation de plusieurs étapes d'ingénierie de commande. Pour ce faire, AML définit la RoleClass AML "Facet" (voir 6.4.4). Des dispositions normatives sont spécifiées en 8.3.

Le sous-groupe décrit d'attributs et d'interfaces est lié à un certain aspect technique et peut archiver des informations concernant les solutions ou les modèles techniques correspondants. La syntaxe ou la sémantique de ces noms ou valeurs d'attributs ne fait pas partie intégrante de la présente partie de la CEI 62714 et est interprétée par un outil technique externe qui connaît la syntaxe et la sémantique des informations correspondantes. Par conséquent, ces algorithmes requièrent uniquement les informations nécessaires concernant Facet pour l'exécution des tâches techniques automatisées. Par exemple, considérer que les attributs d'un objet comprennent un nom de modèle de code PLC et que les interfaces décrivent des entrées ou des sorties de ou vers ce modèle. Ainsi, un algorithme de génération de codes PLC, qui connaît la sémantique de ces attributs et interfaces, peut générer un code PLC à partir de ces informations. La même opération est possible avec les modèles IHM. Les algorithmes externes mentionnés ou la sémantique des attributs ou interfaces correspondants ne relèvent pas du domaine d'application de la présente partie de la CEI 62714. En association avec le concept AML Group, une automatisation des étapes techniques est possible.

A.2.3.2 Exemple

La Figure A.14 explicite le concept AML Facet à l'aide d'un exemple: l'objet "Conveyor1" comprend les attributs “A” et “B” ainsi que les interfaces “X” et “Y”. L'objet Facet attribué “PLCFacet” fait référence à l'attribut “A” et l'interface “X”, tandis que l'objet Facet attribué “HMIFacet” fait référence aux attributs “A” et “B”, ainsi que l'interface “Y”. De fait, les deux objets Facet offrent une vision filtrée de certaines informations techniques pertinentes pour différentes tâches techniques.

Cas d'utilisation: L'attribut "A" peut être le nom d'un modèle de code PLC spécifique au fournisseur, qui décrit la fonctionnalité de l'objet “Conveyor1”. L'interface "X" peut être le nom d'un signal d'entrée requis pour ce modèle de code. L'attribut “B” peut être le nom d'un modèle d'IHM spécifique pour le transporteur et l'interface “Y” peut être un signal qu'il convient de présenter sur l'IHM. Ces informations permettent à un générateur de PLC ou IHM de générer des solutions de manière automatique. Un exemple est décrit en A.2.4.4.



Légende

Anglais	Français
Attribute	Attribut

Figure A.14 – Exemple d'AML Facet

```
<InstanceHierarchy Name="FacetExample">
  <InternalElement Name="Conveyor1" ID="GUID1">
    <Attribute Name="A"/>
    <Attribute Name="B"/>
    <ExternalInterface Name="X"/>
    <ExternalInterface Name="Y"/>
    <InternalElement Name="PLCFacet" ID="GUID2">
      <Attribute Name="A"/>
      <ExternalInterface Name="X"/>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Facet"/>
    </InternalElement>
    <InternalElement Name="HMIFacet" ID="GUID3">
      <Attribute Name="A"/>
      <Attribute Name="B"/>
      <ExternalInterface Name="Y"/>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Facet"/>
    </InternalElement>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
  </InternalElement>
</InstanceHierarchy>
```

Figure A.15 – Texte XML de l'exemple d'AML Facet

A.2.4 Concept AML Group

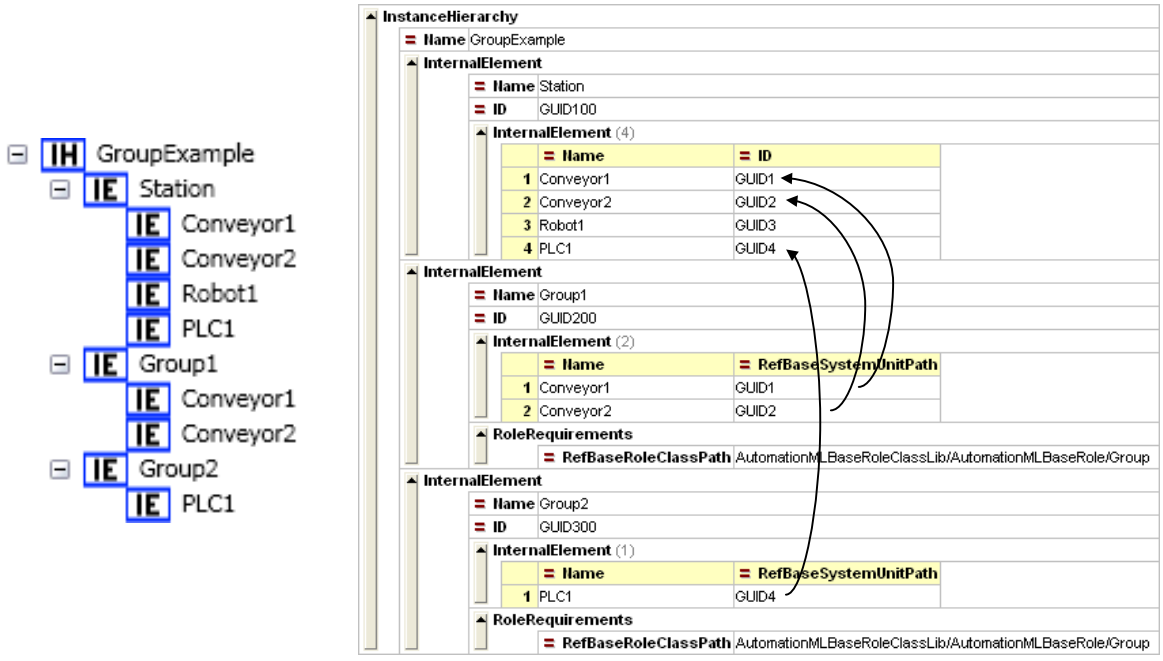
A.2.4.1 Description du concept

Le concept AML Group permet de séparer les informations de structure des informations d'instance. Dans la mesure où différents outils techniques dans un environnement d'outils hétérogène peuvent requérir différentes visions des mêmes données, il peut se révéler utile d'archiver ces visions séparément. Ceci est possible grâce au concept AML Group, et permet de structurer des objets identiques dans différentes hiérarchies.

La définition de l'attribut de groupe „AssociatedFacet“ permet d'associer un groupe avec un type de Facettes caractérisées par un nom unique. Ceci permet aux algorithmes techniques externes d'identifier automatiquement les objets associés et leurs facettes correspondantes afin de déduire les informations techniques. Pour ce faire, AML définit la RoleClass AML "Group" (voir 6.4.3). Des dispositions normatives sont spécifiées en 8.4.

A.2.4.2 Exemple

La Figure A.16a) décrit le concept Group au moyen d'une structure “Station” qui contient les objets “Conveyor1”, “Conveyor2”, “Robot1” et “PLC1”. De plus, les objets “Group1” et “Group2” décrivent les mêmes données dans différentes hiérarchies: “Group1” fournit une vision structurelle des transporteurs uniquement, tandis que “Group2” illustre uniquement les objets de type PLC. Conformément au A.2.14 de la CEI 62424:2008, le format CAEX prévoit l'archivage de ce type de structures croisées. La Figure A.16b) illustre une mise en œuvre AML de cet exemple et la Figure A.17 fournit le texte XML correspondant. La combinaison entre le concept Facet et le concept Port est décrite en A.2.4.3.



a) Hiérarchie **b) Vue en tableau**

Figure A.16 – Exemple d'AML Group

```
<InstanceHierarchy Name="GroupExample">
  <InternalElement Name="Station" ID="GUID100">
    <InternalElement Name="Conveyor1" ID="GUID1"/>
    <InternalElement Name="Conveyor2" ID="GUID2"/>
    <InternalElement Name="Robot1" ID="GUID3"/>
    <InternalElement Name="PLC1" ID="GUID4"/>
  </InternalElement>
  <InternalElement Name="Group1" ID="GUID200">
    <InternalElement Name="Conveyor1" RefBaseSystemUnitPath="GUID1"/>
    <InternalElement Name="Conveyor2" RefBaseSystemUnitPath="GUID2"/>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Group"/>
  </InternalElement>
  <InternalElement Name="Group2" ID="GUID300">
    <InternalElement Name="PLC1" RefBaseSystemUnitPath="GUID4"/>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Group"/>
  </InternalElement>
</InstanceHierarchy>
```

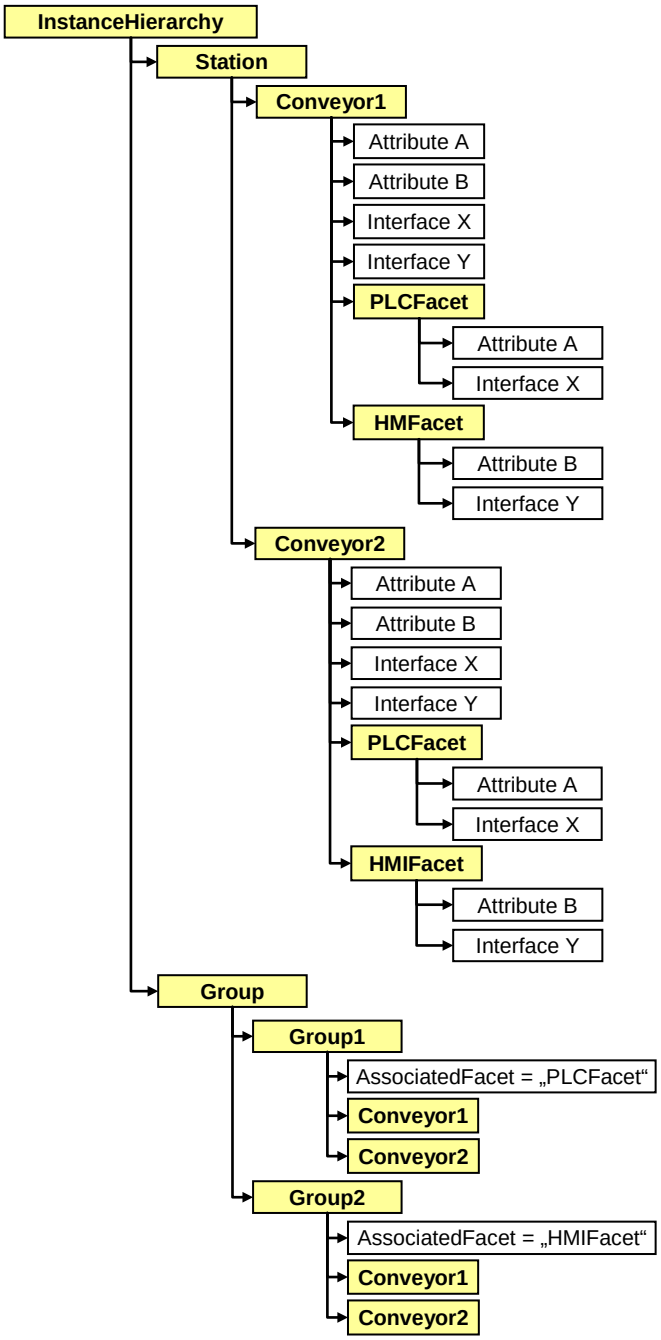
Figure A.17 – Texte XML de l'exemple d'AML Group

A.2.4.3 Combinaison des concepts Group et Facet

La Figure A.18 présente un exemple de combinaison des concepts Group et Facet. L'élément InstanceHierarchy présenté illustre un objet "Station" AML qui comprend les objets AML “Conveyor1” et “Conveyor2”. Chacun de ces transporteurs comprend deux attributs et deux interfaces.

L'objet AML "Group" présente les groupes imbriqués **“**Group1” et “Group2”. Ces deux groupes font référence aux objets de transporteur, mais comportent des associations de facettes différentes.

Cas d'utilisation: Un algorithme de génération de codes de commande peut fonctionner dans l'élément InstanceHierarchy, en identifiant tous les groupes associés à un élément "PLCFacet", puis générer le code en évaluant les objets référencés.



Légende

Anglais	Français
Station	Poste
Attribute	Attribut
Group	Groupe
Interface	Interface

Figure A.18 – Combinaison des concepts Facet et Group

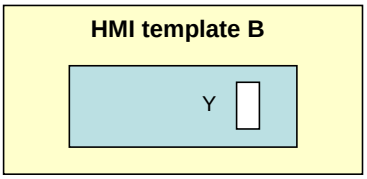
La Figure A.19 présente le texte XML correspondant associé à l'exemple présenté à la Figure A.18.

```
<InstanceHierarchy Name="FacetGroupCombination">
  <InternalElement Name="Conveyor1" ID="GUID1">
    <Attribute Name="A"/>
    <Attribute Name="B"/>
    <ExternalInterface Name="X"/>
    <ExternalInterface Name="Y"/>
    <InternalElement Name="PLCFacet" ID="GUID2">
      <Attribute Name="A"/>
      <ExternalInterface Name="X"/>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Facet"/>
    </InternalElement>
    <InternalElement Name="HMIFacet" ID="GUID3">
      <Attribute Name="B"/>
      <ExternalInterface Name="Y"/>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Facet"/>
    </InternalElement>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource"/>
  </InternalElement>
  <InternalElement Name="Conveyor2" ID="GUID4">
    <Attribute Name="A"/>
    <Attribute Name="B"/>
    <ExternalInterface Name="X"/>
    <ExternalInterface Name="Y"/>
    <InternalElement Name="PLCFacet" ID="GUID5">
      <Attribute Name="A"/>
      <ExternalInterface Name="X"/>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Facet"/>
    </InternalElement>
    <InternalElement Name="HMIFacet" ID="GUID6">
      <Attribute Name="B"/>
      <ExternalInterface Name="Y"/>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Facet"/>
    </InternalElement>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource"/>
  </InternalElement>
  <InternalElement Name="Group" ID="GUID7">
    <InternalElement Name="Group1" ID="GUID8">
      <Attribute Name="AccociatedFacet">
        <Value>PLCFacet</Value>
      </Attribute>
      <InternalElement Name="Conveyor1" RefBaseSystemUnitPath="GUID1"/>
      <InternalElement Name="Conveyor2" RefBaseSystemUnitPath="GUID2"/>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Group"/>
    </InternalElement>
    <InternalElement Name="Group2" ID="GUID9">
      <Attribute Name="AccociatedFacet">
        <Value>HMIFacet</Value>
      </Attribute>
      <InternalElement Name="Conveyor1" RefBaseSystemUnitPath="GUID1"/>
      <InternalElement Name="Conveyor2" RefBaseSystemUnitPath="GUID2"/>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Group"/>
    </InternalElement>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Group"/>
  </InternalElement>
</InstanceHierarchy>
```

Figure A.19 – Vue de texte XML de l'exemple combiné des concepts Facet et Group

A.2.4.4 Génération automatique d'une IHM en utilisant les concepts Group et Facet

On suppose, sur la base de l'exemple fourni, que l'attribut des transporteurs "B" représente un modèle d'IHM qui visualise la variable "Y". La Figure A.20 illustre ce modèle d'IHM générique d'un transporteur.



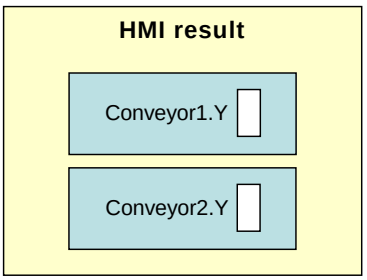
Légende

Anglais	Français
HMI template	Modèle IHM

Figure A.20 – Modèle d'IHM générique "B" visualisant une variable de processus "Y" d'un transporteur

Sur la base des instances concrètes des transporteurs, un algorithme technique est activé afin d'identifier que l'objet AML "Group2" est associé à l'attribut HMI Facet. Cet algorithme identifie dans ce cas que les instances “Conveyor1” et “Conveyor2” font partie intégrante de l'IHM. L'algorithme peut extraire les informations de type IHM de chacun des deux transporteurs, peut identifier le modèle d'IHM correspondant et peut associer les signaux corrects à visualiser.

La Figure A.21 représente l'IHM obtenue.



Légende

Anglais	Français
HMI result	Résultat IHM

Figure A.21 – Résultat généré "B" de l'IHM visualisant les deux transporteurs avec des variables de processus individuelles

A.2.5 Concept PropertySet

A.2.5.1 Description du concept

Un PropertySet agit comme une taxonomie d'attribut, un dictionnaire structuré. Il est modélisé en tant que classe de rôle contenant des attributs prédéfinis qui décrivent les propriétés d'un certain domaine d'application et est déduit de la classe de rôle standard “PropertySet” (voir 6.4.13). Il comprend une liste d'attributs à convention syntaxique et sémantique. Les ensembles de propriétés sont regroupés dans des bibliothèques de classes de rôles.

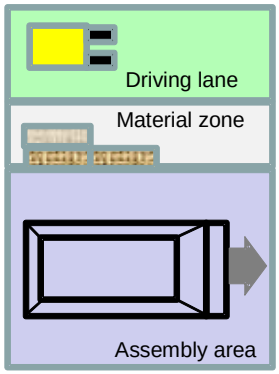
Les objets AML peuvent être associés à un ou plusieurs ensembles de propriétés. Pour chaque ensemble de propriétés, il convient de créer un objet enfant distinct de type InternalElement CAEX et de l'attribuer à la classe PropertySet correspondante par la définition de ses RoleRequirements. Des ensembles de propriétés multiples peuvent être associés au moyen des multiples InternalElements enfants de l'objet AML correspondant.

L'objet de mapping CAEX permet la mise en correspondance des attributs propriétaires de l'objet AML avec les attributs prédéfinis de manière sémantique de l'objet PropertySet. Ceci permet au logiciel importateur d'interpréter automatiquement ces attributs et de les mettre en correspondance avec les attributs cibles spécifiques aux outils. Ceci simplifie l'échange de données automatique entre différents outils.

Une bibliothèque AML prédéfinie d'ensembles de propriétés peut être spécifiée. Des dispositions normatives sont spécifiées en 8.5.

A.2.5.2 Exemple

A titre d'exemple, le langage AML permet le transfert de l'aménagement d'une chaîne de montage avec plusieurs postes d'assemblage (voir Figure A.22). Le poste d'assemblage comporte différentes zones, à savoir une pour le transport des matériaux, une pour le stockage de ces mêmes matériaux et une pour le montage; comme indiqué ci-dessous. L'outil technique source définit ces zones au moyen d'attributs définis par l'utilisateur, attribués à l'objet, qui représentent le poste.

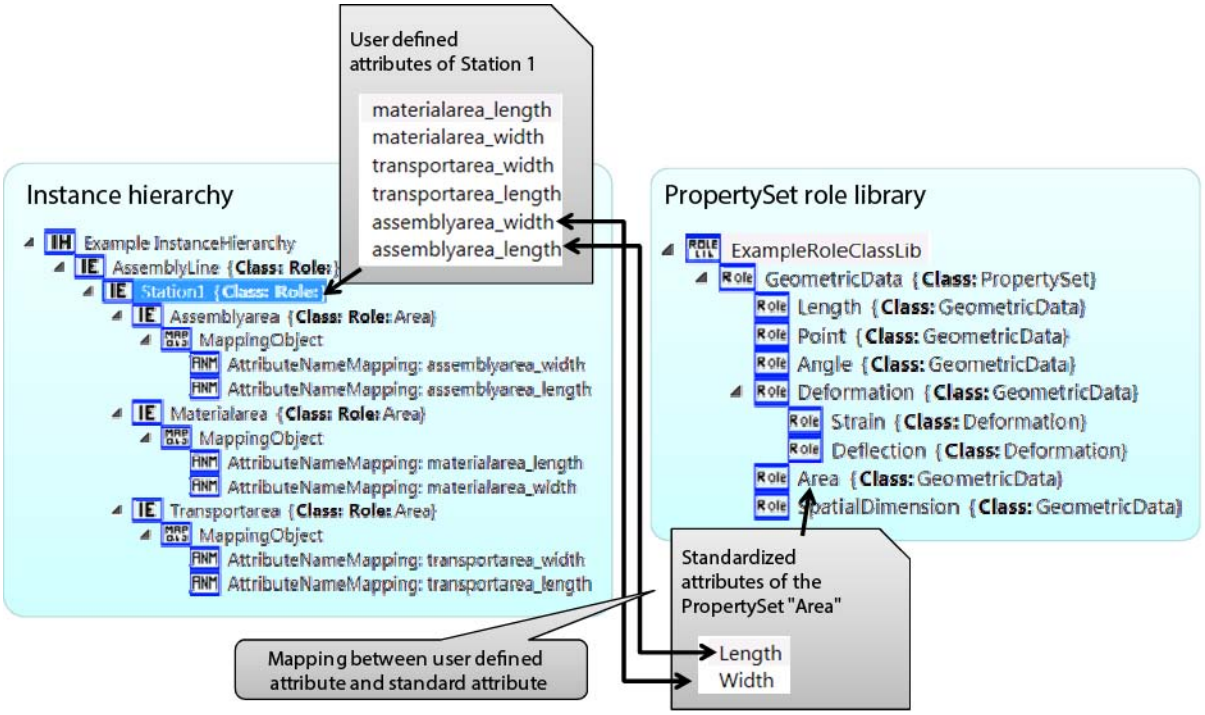


Légende

Anglais	Français
Driving lane	Voie de conduite
Material zone	Zone de stockage des matériaux
Assembly area	Zone de montage

Figure A.22 – Exemple de PropertySet

La Figure A.23 illustre le modèle AML correspondant. La partie gauche de la Figure illustre la hiérarchie d'instances pour l'attribut Station1 de modélisation des trois zones. Chacune de ces zones comporte un ensemble d'attributs définis par l'utilisateur et une référence au PropertySet “Area”. Le texte XML correspondant est présenté à la Figure A.24.



Légende

Anglais	Français
User defined Attributes of Station 1	Attributs définis par l'utilisateur du poste 1
Instance hierarchy	Hiérarchie d'instances
PropertySet role library	Bibliothèque de rôles PropertySet
Mapping between user defined attribute and standard attribute	Mapping entre l'attribut défini par l'utilisateur et l'attribut standard
Standardized attributes of the PropertySet "Area"	Attributs normalisés du PropertySet "Area"
Length	Longueur
Width	Largeur

Figure A.23 – Exemple de PropertySet

```
<InstanceHierarchy Name="Example InstanceHierarchy">
  <InternalElement Name="AssemblyLine" ID="{8b2510b4-7fc5-4f54-9d09-a3197a062604}">
    <InternalElement Name="Station1" ID="{54500138-c6a1-47c8-80c9-bee35662563c}">
      <Attribute Name="materialarea_length" />
      <Attribute Name="materialarea_width" />
      <Attribute Name="transportarea_width" />
      <Attribute Name="transportarea_length" AttributeDataType="xs:double" />
      <Attribute Name="assemblyarea_width" AttributeDataType="xs:double" />
      <Attribute Name="assemblyarea_length" AttributeDataType="xs:double" />
      <InternalElement Name="Assemblyarea" ID="{e08f53e5-eb0f-45c8-b42f-4714824dd05c}">
        <RoleRequirements RefBaseRoleClassPath="ExampleRoleClassLib/GeometricData/Area" />
        <MappingObject>
          <AttributeNameMapping SystemUnitAttributeName="assemblyarea_width" RoleAttributeName="Width" />
          <AttributeNameMapping SystemUnitAttributeName="assemblyarea_length" RoleAttributeName="Length" />
        </MappingObject>
      </InternalElement>
      <InternalElement Name="Materialarea" ID="{668360af-d108-4b60-be53-ddb262bc470e}">
        <RoleRequirements RefBaseRoleClassPath="ExampleRoleClassLib/GeometricData/Area" />
        <MappingObject>
          <AttributeNameMapping SystemUnitAttributeName="materialarea_length" RoleAttributeName="Length" />
          <AttributeNameMapping SystemUnitAttributeName="materialarea_width" RoleAttributeName="Width" />
        </MappingObject>
      </InternalElement>
      <InternalElement Name="Transportarea" ID="{19418967-cd7c-46ea-adea-81aa52f6b685}">
        <RoleRequirements RefBaseRoleClassPath="ExampleRoleClassLib/GeometricData/Area" />
        <MappingObject>
          <AttributeNameMapping SystemUnitAttributeName="transportarea_width" RoleAttributeName="Width" />
          <AttributeNameMapping SystemUnitAttributeName="transportarea_length" RoleAttributeName="Length" />
        </MappingObject>
      </InternalElement>
    </InternalElement>
  </InternalElement>
</InstanceHierarchy>
```

Figure A.24 – Texte XML pour la hiérarchie d'instances

La partie droite de la Figure A.24 illustre une bibliothèque AML PropertySet échantillon. Cette bibliothèque AML de RoleClass contient la RoleClass “GeometricData”, déduite du “PropertySet” Base-RoleClass. La RoleClass “GeometricData” proprement dite comporte des dérivations supplémentaires, qui définissent certaines propriétés géométriques de base. La RoleClass appelée "Area" définit les attributs “Length” et “Width” comme étant des attributs de Type “double” et Unit “m [metre]”. Le texte XML de la Figure A.25 présente les définitions des attributs de ces PropertySets.

```
- <RoleClass Name="GeometricData"
    RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/PropertySet
  - <RoleClass Name="Length"
      RefBaseClassPath="ExampleRoleClassLib/GeometricData">
      <Attribute Name="lengthValue" AttributeDataType="xs:double"
        Unit="m" />
    </RoleClass>
  - <RoleClass Name="Point"
      RefBaseClassPath="ExampleRoleClassLib/GeometricData">
      <Attribute Name="xAxis" AttributeDataType="xs:double" />
      <Attribute Name="yAxis" AttributeDataType="xs:double" />
      <Attribute Name="zAxis" AttributeDataType="xs:double" />
    </RoleClass>
  - <RoleClass Name="Angle"
      RefBaseClassPath="ExampleRoleClassLib/GeometricData">
      <Attribute Name="angleValue" AttributeDataType="xs:double"
        Unit="rad" />
    </RoleClass>
  - <RoleClass Name="Deformation"
      RefBaseClassPath="ExampleRoleClassLib/GeometricData">
      <Description>Deformation of the material is the change in geometry
        when stress is applied (in the form of force loading, gravitational
        field, acceleration, thermal expansion, etc.). Deformation is
        expressed by the displacement field of the material</Description>
      <Attribute Name="deformationValue" AttributeDataType="xs:double" />
    - <RoleClass Name="Strain"
        RefBaseClassPath="ExampleRoleClassLib/GeometricData/Deformation">
        <Description>Strain is the deformation per unit length</Description>
      </RoleClass>
    - <RoleClass Name="Deflection"
        RefBaseClassPath="ExampleRoleClassLib/GeometricData/Deformation">
        <Description>Deflection is a term to describe the magnitude to
          which a structural element bends under a load</Description>
      </RoleClass>
    </RoleClass>
  - <RoleClass Name="Area"
      RefBaseClassPath="ExampleRoleClassLib/GeometricData">
      <Attribute Name="Length" AttributeDataType="xs:double" Unit="m" />
      <Attribute Name="Width" AttributeDataType="xs:double" Unit="m" />
    </RoleClass>
  - <RoleClass Name="SpatialDimension"
      RefBaseClassPath="ExampleRoleClassLib/GeometricData">
      <Attribute Name="XAxis" AttributeDataType="xs:double" Unit="m" />
      <Attribute Name="YAxis" AttributeDataType="xs:double" Unit="m" />
      <Attribute Name="ZAxis" AttributeDataType="xs:double" Unit="m" />
    </RoleClass>
  </RoleClass>
```

Figure A.25 – Exemple de bibliothèque AML de PropertySet sous forme de code XML

Les attributs PropertySets permettent à l'outil exportateur d'archiver les objets définis par l'utilisateur avec des attributs également définis par l'utilisateur, et d'expliquer la sémantique de ces attributs définis par l'utilisateur à l'aide de mappings. Ceci prend en charge l'interprétation automatique de ces attributs. Au final, un outil technique cible peut interpréter les données et effectuer les conversions d'unités dans les unités requises des attributs PropertySet.

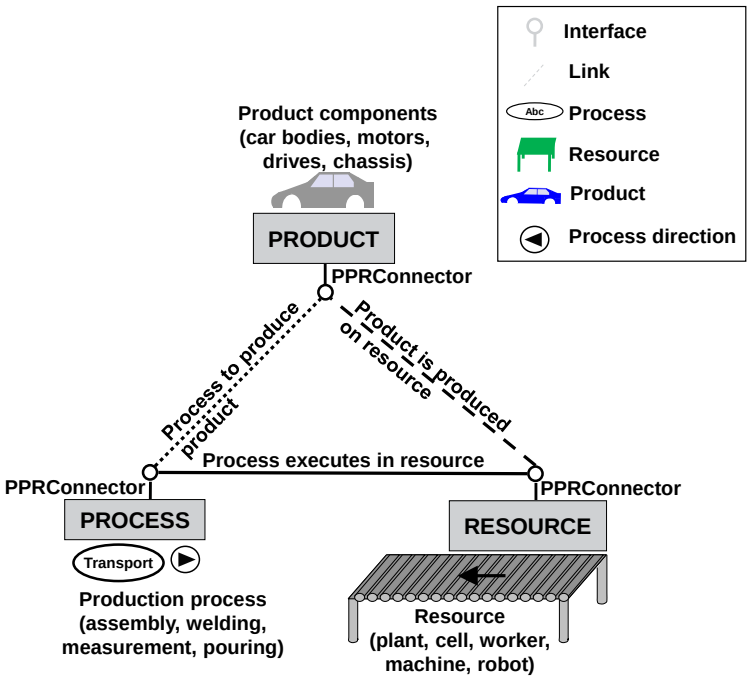
A.2.6 Concept Process-Product-Resource (Processus-Produit-Ressource)

A.2.6.1 Description du concept

Dans le cadre de la structuration de données complexes d'ingénierie d'usine, la répartition des données en trois parties que sont les ressources, les processus et les produits, s'est avérée efficace dans la pratique. Ce concept est appliqué dans des domaines différents, par exemple, pour les outils techniques numériques, ou dans le cadre de la CEI 62264, au niveau du système d'exécution de fabrication (MES).

- Dans une perspective centrée sur les ressources, ces dernières constituent le composant central du modèle: elles exécutent les processus et manipulent les produits. Dans le langage AML, une ressource est une entité impliquée dans la production, y compris les installations, les robots, les machines, leur état, le matériel, les messages potentiels, etc. Selon ces éléments, les ressources peuvent être les composants matériels d'un système de production, ainsi que les systèmes logiciels, par exemple, systèmes SCADA. Dans le langage AML, les ressources sont typiquement modélisées dans une hiérarchie d'installation formant la topologie de l'installation.
- Dans une perspective centrée sur le produit, le produit fabriqué est la cible de toutes les attentions. Il détermine quels sont les processus qu'il convient d'appliquer aux matériaux ou aux produits intermédiaires, et quels sont les matériels qu'il convient par conséquent d'utiliser. Ces dispositions sont valides dans le domaine du contrôle continu, discret ou de lots. Un produit au format AML illustre un bien fabriqué. Il peut être élaboré de manière hiérarchique. Il est essentiel qu'il ne soit pas nécessaire que les produits soient des produits finaux. Les résultats d'essai relèvent des produits, ainsi que les données de produits et les documents correspondants.
- Dans une perspective centrée sur les processus, ces derniers constituent les éléments centraux du modèle. Un processus dans le langage AML représente un processus de fabrication y compris les sous-processus. Les paramètres de processus, la chaîne de processus et la planification de processus font partie intégrante des processus. En termes techniques, les processus modifient les produits. Ceci correspond à l'utilisation dans le langage AML dans la mesure où les produits finaux sont fabriqués à partir de différents sous-produits, ou les traitements chimiques modifient les substances. Toutefois, les processus sont associés (par des relations) aux ressources et inversement.

Dans tous les cas, les représentations des ressources, produits et processus sont liées entre elles (voir Figure A.26). La ressource "transporteur" ("conveyor") constitue une attribution raisonnable au processus "transport". Un attribut "presse" ("press") peut générer des "volumes de déchets" ("scrap cubes"). Et un procédé "de soudage" ("welding") peut "souder" ("weld") deux "métaux" ("metals") entre eux.



Légende

Anglais	Français
Interface	Interface
Link	Liaison

Anglais	Français
Process	Processus
Resource	Ressource
Product	Produit
Process direction	Sens du processus
Product components (car bodies, motors, drives, chassis)	Composants de produits (carrosseries, moteurs, transmissions, châssis)
PRODUCT	PRODUIT
Process to produce product	Processus de fabrication du produit
Product is produced on resource	Fabrication du produit avec les ressources
Process executes in resource	Exécution du processus par les ressources
PROCESS	PROCESSUS
RESOURCE	RESSOURCE
Production process (assembly, welding, measurement, pouring)	Processus de fabrication (assemblage, soudage, mesurage, coulage)
Resource (plant, cell, worker, machine, robot)	Ressource (installation, cellule, travailleur, machine, robot)

Figure A.26 – Éléments de base du concept Process-Product-Resource

La création d'une liaison entre ces éléments exige une interface. Pour ce faire, AML définit la classe d'interface standard PPRConnector (voir Figure A.27). Des dispositions normatives concernant le PPRConnector sont spécifiées en 6.3.5. Cette interface permet d'établir des liaisons entre les éléments au moyen des InternalLinks CAEX standard (voir 5.6.6). Ainsi, les ressources peuvent être reliées aux produits qu'elles peuvent manipuler.

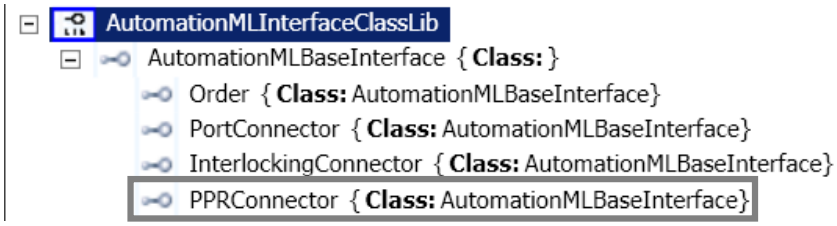
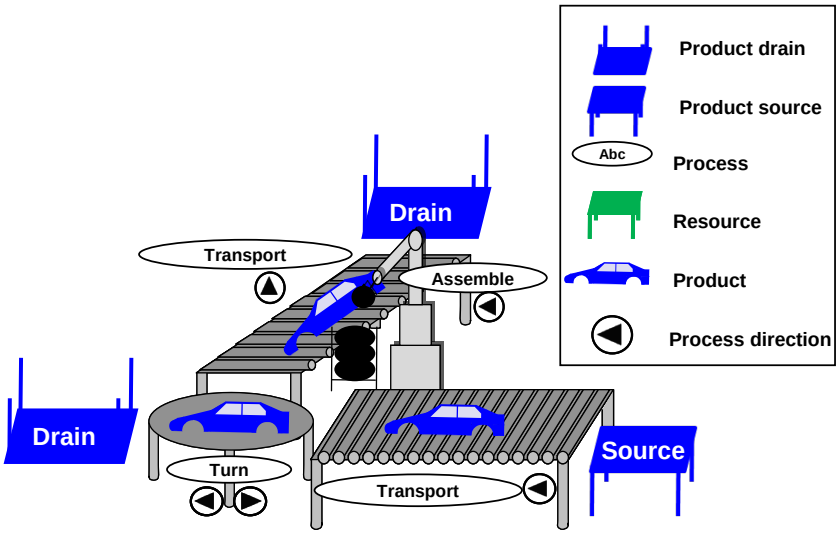


Figure A.27 – Interface PPRConnector

A.2.6.2 Exemple

L'exemple suivant (voir Figure A.28) illustre l'application de ce concept avec AML. Il consiste en deux transporteurs (C1 et C2), une plaque tournante (TT1) et un robot (RB1). Ces éléments constituent les ressources de l'installation. Le robot assemble les roues des véhicules. Les roues, ainsi que les véhicules, sont des produits. Les processus de fabrication, dans cet exemple, sont le transport, la rotation et l'assemblage.



Légende

Anglais	Français
Product drain	Evacuation du produit
Product source	Source du produit
Process	Processus
Resource	Ressource
Product	Produit
Process direction	Sens du processus
Assemble	Assemblage
Turn	Rotation
Drain	Evacuation

Figure A.28 – Exemple de concept Process-Product-Resource

Dans le langage AML, le concept Process-Product-Resource est modélisé au moyen du concept de rôle CAEX (voir la CEI 62424:2008.A.2.9) et des relations entre les éléments (voir 5.6.6). Les ensembles d'éléments comportant les rôles attribués “Resource”, “Product” ou “Process” s'excluent mutuellement par paire. Cela signifie qu'une ressource ne peut être simultanément un produit ou un processus. Les classes de rôles correspondantes font partie intégrante de la bibliothèque AutomationMLBaseRoleClassLib (voir Figure A.29 et 6.4).

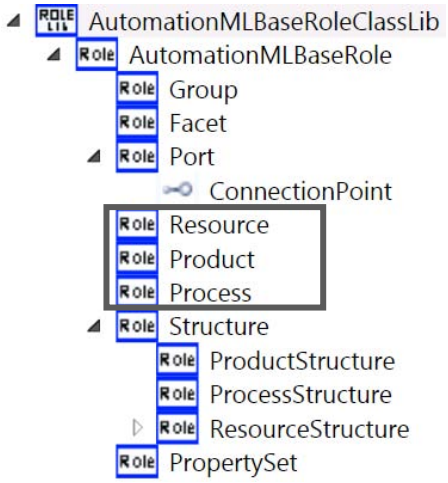
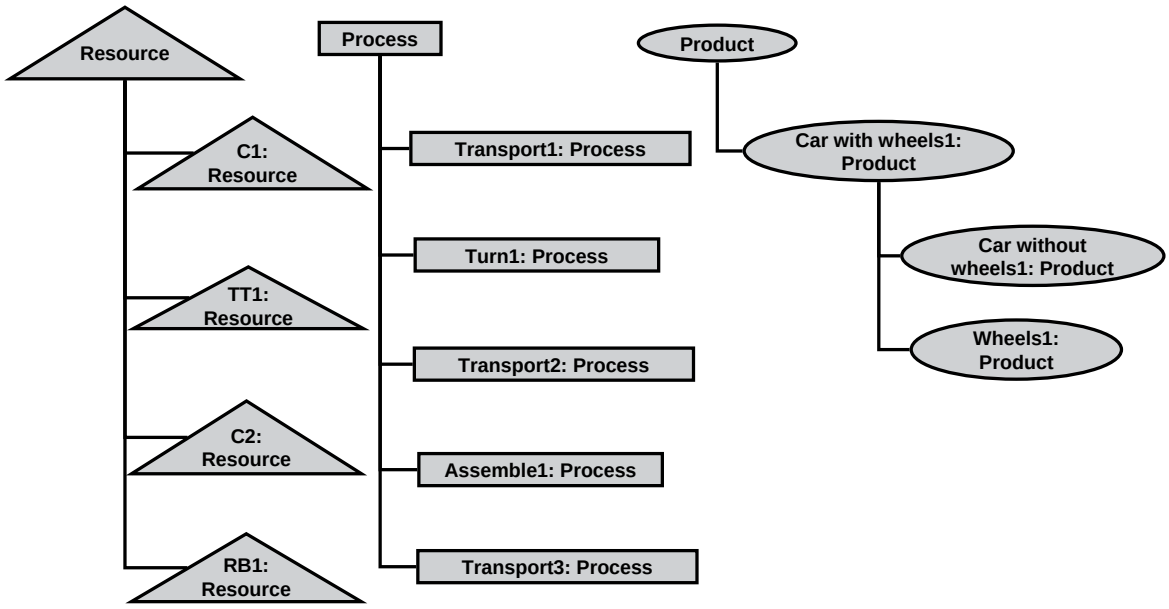


Figure A.29 – Rôles AML requis pour le concept Process-Product-Resource

Dans l'exemple (voir Figure A.28), le rôle “Resource” est attribué aux transporteurs, au robot et à la plaque tournante. Les véhicules et les roues sont attribués au rôle “Product” et le rôle “Process” est attribué aux éléments de processus de transport, de rotation et d'assemblage. Tous les éléments sont archivés dans la sous-arborescence correspondante que l'on peut observer à la Figure A.30. L'ordre des processus, produits ou ressources peut être exprimé de manière explicite par les liaisons entre les interfaces correspondantes de type "Order" (ceci n'est pas illustré dans cet exemple pour des raisons de lisibilité).

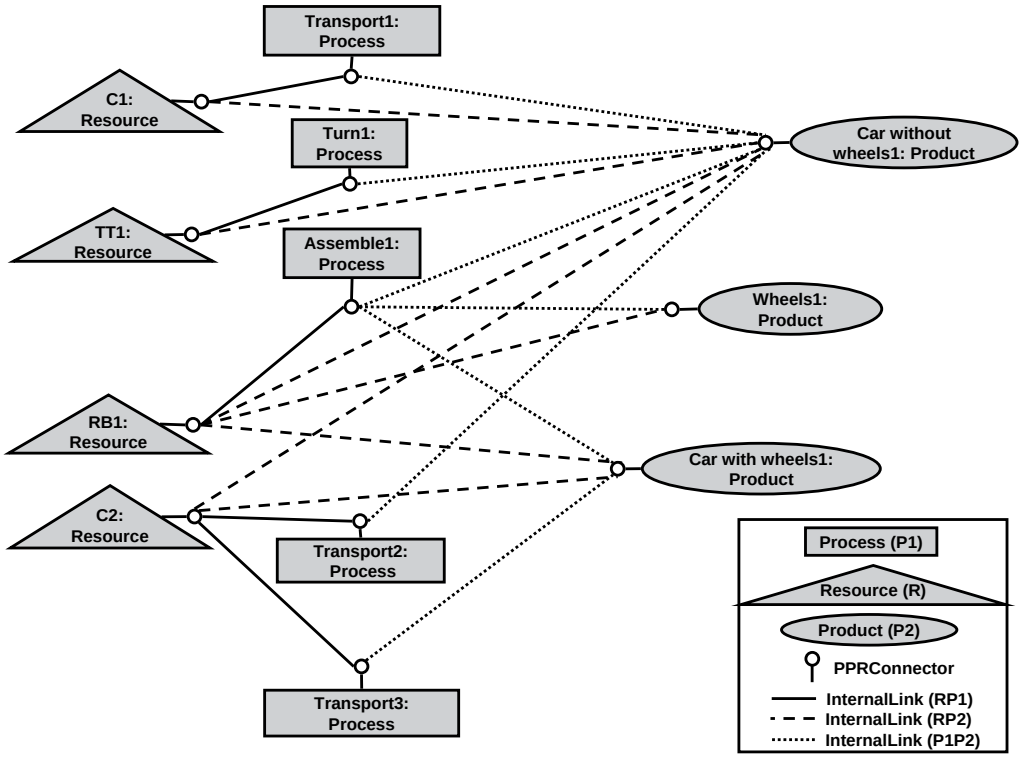


Légende

Anglais	Français
Resource	Ressource
Process	Processus
Product	Produit
Car with wheels1	Véhicule avec roues1
Car without wheels1	Véhicule sans roues1

Figure A.30 – Éléments de l'exemple

Chaque élément illustré dans l'exemple a une interface PPRConnector. Les liaisons complètes de l'exemple sont représentées à la Figure A.31. Les traits continus représentent les liaisons entre les ressources et les processus, les pointillés représentent les liaisons entre les processus et les produits, et les traits discontinus constituent les liaisons entre les ressources et les produits. Cela révèle la complexité. Ainsi, les connexions redondantes peuvent être omises.

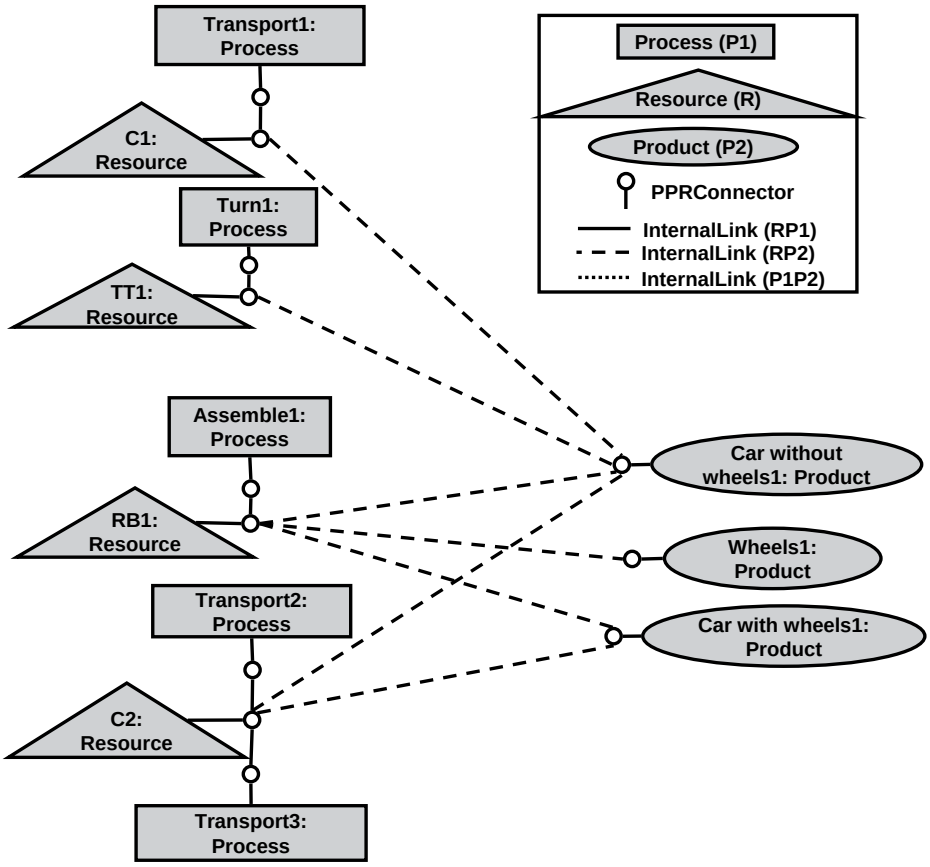


Légende

Anglais	Français
Process	Processus
Resource	Ressource
Car without wheels1	Véhicule sans roues1
Product	Produit
Car with wheels1	Véhicule avec roues1

Figure A.31 – Liaisons de l'exemple

La Figure A.32 présente la perspective centrée sur les ressources de l'exemple considéré. Par conséquent, seules douze liaisons, et non dix-neuf, sont nécessaires. Le transporteur “C1” est connecté au produit “Car without wheels1“ tout comme la plaque tournante “TT1” et le transporteur “C2”. Dans la mesure où le robot assemble les roues des véhicules sur le transporteur “C2”, le robot “RB1” est connecté aux éléments “Car without wheels1”, “Car with wheels1” et “Wheels1”. De plus, le transporteur “C2” est connecté à l'élément “Car with wheels1”. Le processus “Transport1” est attribué à l'élément “C1”, et les éléments “Transport2” et “Transport3” sont connectés au transporteur “C2”. L'élément “Assemble1” est lié au robot “RB1” et l'élément “Turn1” est lié à la plaque tournante “TT1”. Les liaisons entre les produits et les processus (pointillés à la Figure A.31) peuvent être déduites des liaisons existantes. Le modèle peut faire l'objet d'une rotation arbitraire et être disposé de manière à centrer les éléments du produit ou processus de type.



Légende

Anglais	Français
Process	Processus
Resource	Ressource
Product	Produit
Car without wheels1	Véhicule sans roues1
Car with wheels1	Véhicule avec roues1

Figure A.32 – Liaisons de la perspective centrée sur les ressources dans l'exemple

La Figure A.33 illustre l'arborescence d'objets AML, y compris une liaison entre le transporteur "C1" et le processus “Transport1“.

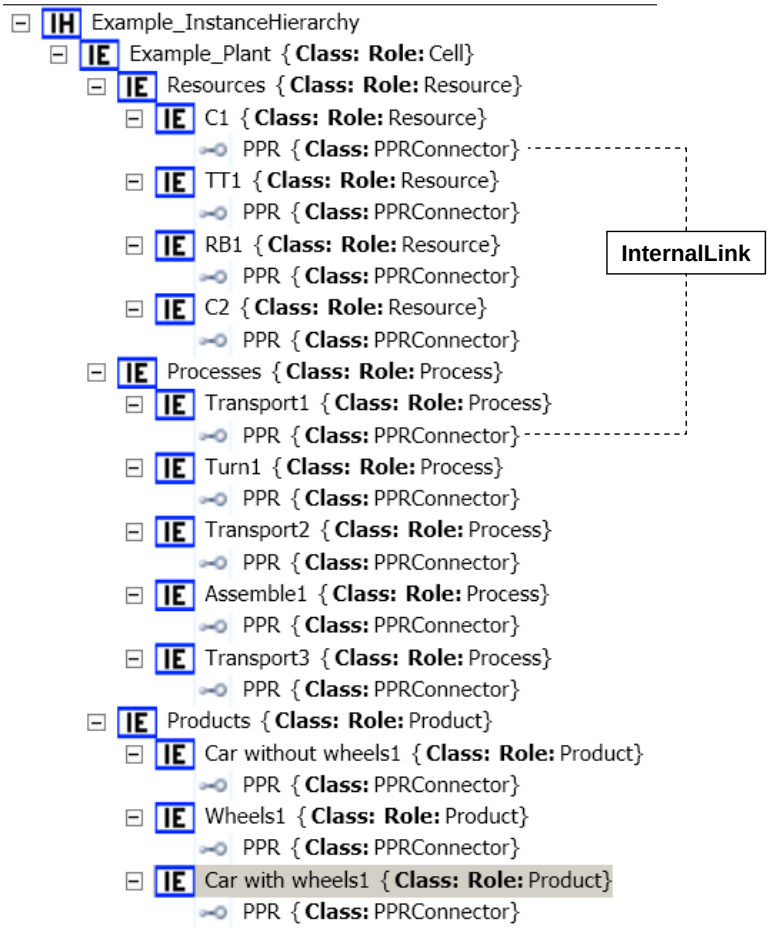
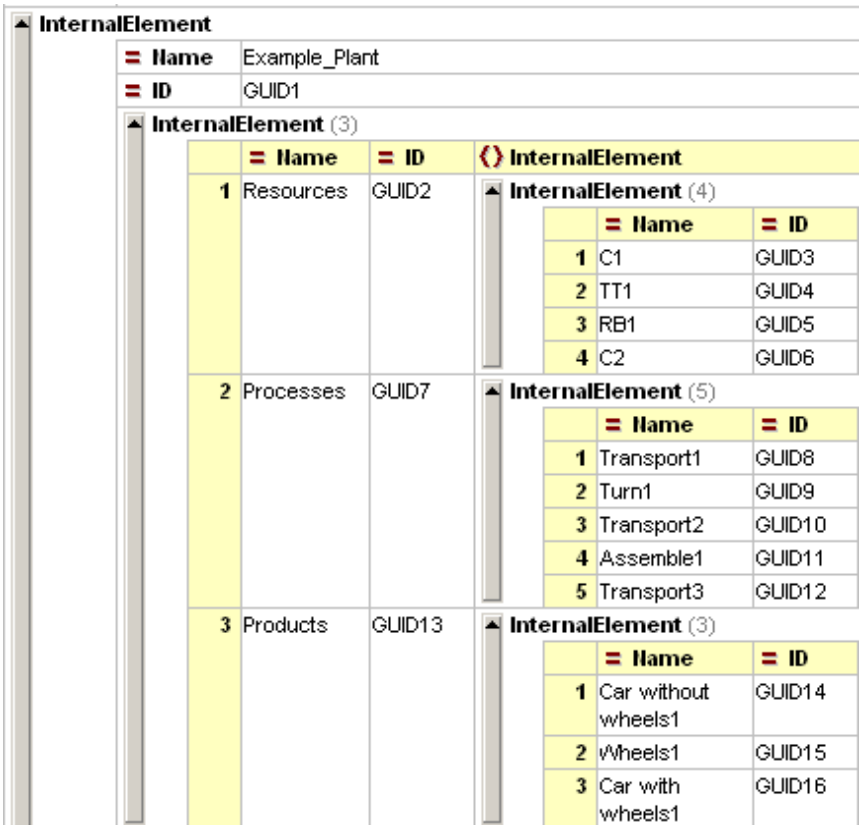


Figure A.33 – InstanceHierarchy de l'exemple en langage AML

Le modèle XML correspondant est illustré à la Figure A.35. Le premier niveau de l'exemple comporte les trois éléments de base, à savoir: “Resources”, “Processes” et “Products” modélisés comme un InternalElement CAEX.



InternalElement

= Name	Example_Plant
= ID	GUID1

InternalElement (3)

	= Name	= ID	InternalElement
1	Resources	GUID2	InternalElement (4)
2	Processes	GUID7	InternalElement (5)
3	Products	GUID13	InternalElement (3)

InternalElement (4)

	= Name	= ID
1	C1	GUID3
2	TT1	GUID4
3	RB1	GUID5
4	C2	GUID6

InternalElement (5)

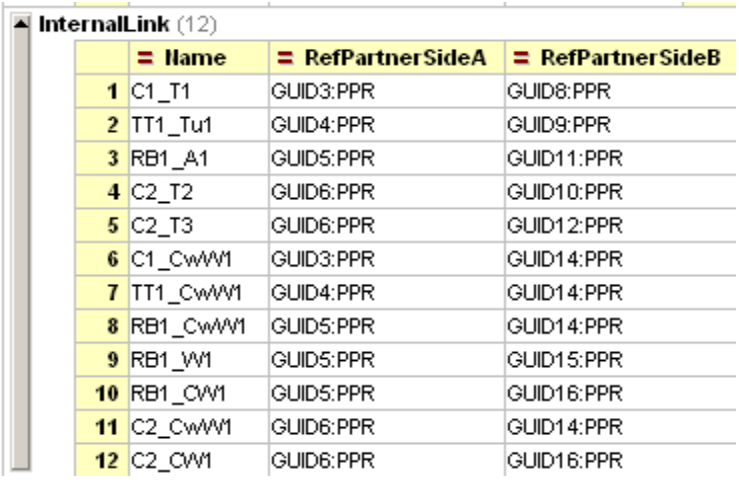
	= Name	= ID
1	Transport1	GUID8
2	Turn1	GUID9
3	Transport2	GUID10
4	Assemble1	GUID11
5	Transport3	GUID12

InternalElement (3)

	= Name	= ID
1	Car without wheels1	GUID14
2	Wheels1	GUID15
3	Car with wheels1	GUID16

Figure A.34 – InternalElements de l'exemple

L'objet "Resources" comprend les quatre composants de l'exemple: les deux transporteurs, la plaque tournante et le robot. Ces composants sont également du type InternalElement. Ils comportent une ExternalInterface PPRConnector et attribuent la classe de rôle “Resource”. Les processus et produits comportent également l'attribution d'une interface et d'un rôle. Afin de relier les éléments donnés dans l'exemple, les InternalLinks sont généralement placées au même niveau que l'élément de base le plus important. Les liaisons au format XML sont illustrées à la Figure A.35.



InternalLink (12)

	= Name	= RefPartnerSideA	= RefPartnerSideB
1	C1_T1	GUID3:PPR	GUID8:PPR
2	TT1_Tu1	GUID4:PPR	GUID9:PPR
3	RB1_A1	GUID5:PPR	GUID11:PPR
4	C2_T2	GUID6:PPR	GUID10:PPR
5	C2_T3	GUID6:PPR	GUID12:PPR
6	C1_CwW1	GUID3:PPR	GUID14:PPR
7	TT1_CwW1	GUID4:PPR	GUID14:PPR
8	RB1_CwW1	GUID5:PPR	GUID14:PPR
9	RB1_W1	GUID5:PPR	GUID15:PPR
10	RB1_CW1	GUID5:PPR	GUID16:PPR
11	C2_CwW1	GUID6:PPR	GUID14:PPR
12	C2_CW1	GUID6:PPR	GUID16:PPR

Figure A.35 – InternalLinks de l'exemple

La Figure A.36 donne une vue d’ensemble complète de l'exemple.

```
<InstanceHierarchy Name="Example_InstanceHierarchy">
  <InternalElement Name="Example_Plant" ID="GUID1">
    <InternalElement Name="Resources" ID="GUID2">
      <InternalElement Name="C1" ID="GUID3">
        <ExternalInterface Name="PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
      </InternalElement>
      <InternalElement Name="TT1" ID="GUID4">
        <ExternalInterface Name="PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
      </InternalElement>
      <InternalElement Name="RB1" ID="GUID5">
        <ExternalInterface Name="PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
      </InternalElement>
      <InternalElement Name="C2" ID="GUID6">
        <ExternalInterface Name="PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
      </InternalElement>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
    </InternalElement>
    <InternalElement Name="Processes" ID="GUID7">
      <InternalElement Name="Transport1" ID="GUID8">
        <ExternalInterface Name="PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Process"/>
      </InternalElement>
      <InternalElement Name="Turn1" ID="GUID9">
        <ExternalInterface Name="PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Process"/>
      </InternalElement>
      <InternalElement Name="Transport2" ID="GUID10">
        <ExternalInterface Name="PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Process"/>
      </InternalElement>
      <InternalElement Name="Assemble1" ID="GUID11">
        <ExternalInterface Name="PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Process"/>
      </InternalElement>
      <InternalElement Name="Transport3" ID="GUID12">
        <ExternalInterface Name="PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Process"/>
      </InternalElement>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Process"/>
    </InternalElement>
    <InternalElement Name="Products" ID="GUID13">
      <InternalElement Name="Car without wheels1" ID="GUID14">
        <ExternalInterface Name="PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Product"/>
      </InternalElement>
      <InternalElement Name="Wheels1" ID="GUID15">
        <ExternalInterface Name="PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Product"/>
      </InternalElement>
      <InternalElement Name="Car with wheels1" ID="GUID16">
        <ExternalInterface Name="PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Product"/>
      </InternalElement>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Product"/>
    </InternalElement>
    <InternalLink Name="C1_T1" RefPartnerSideA="GUID3:PPR" RefPartnerSideB="GUID8:PPR"/>
    <InternalLink Name="TT1_Tu1" RefPartnerSideA="GUID4:PPR" RefPartnerSideB="GUID9:PPR"/>
    <InternalLink Name="RB1_A1" RefPartnerSideA="GUID5:PPR" RefPartnerSideB="GUID11:PPR"/>
    <InternalLink Name="C2_T2" RefPartnerSideA="GUID6:PPR" RefPartnerSideB="GUID10:PPR"/>
    <InternalLink Name="C2_T3" RefPartnerSideA="GUID6:PPR" RefPartnerSideB="GUID12:PPR"/>
    <InternalLink Name="C1_CwW1" RefPartnerSideA="GUID3:PPR" RefPartnerSideB="GUID14:PPR"/>
    <InternalLink Name="TT1_CwW1" RefPartnerSideA="GUID4:PPR" RefPartnerSideB="GUID14:PPR"/>
    <InternalLink Name="RB1_CwW1" RefPartnerSideA="GUID5:PPR" RefPartnerSideB="GUID14:PPR"/>
    <InternalLink Name="RB1_W1" RefPartnerSideA="GUID5:PPR" RefPartnerSideB="GUID15:PPR"/>
    <InternalLink Name="RB1_CW1" RefPartnerSideA="GUID5:PPR" RefPartnerSideB="GUID16:PPR"/>
    <InternalLink Name="C2_CwW1" RefPartnerSideA="GUID6:PPR" RefPartnerSideB="GUID14:PPR"/>
    <InternalLink Name="C2_CW1" RefPartnerSideA="GUID6:PPR" RefPartnerSideB="GUID16:PPR"/>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Cell"/>
  </InternalElement>
</InstanceHierarchy>
```

Figure A.36 – InstanceHierarchy de l'exemple en langage XML

A.2.7 Prise en charge des rôles multiples

Outre les spécifications de A.2.9 de la CEI 62424:2008, AML définit comment spécifier la prise en charge des rôles multiples pour une instance d'objet. Les rôles multiples sont

intéressants si un objet peut avoir des fonctionnalités multiples. Un exemple qui illustre ces fonctionnalités multiples est un dispositif multifonctionnel ayant la triple fonction conjointe de scanner, imprimante et télécopie. Des dispositions concernant les rôles multiples sont spécifiées en 8.5.

La Figure A.37 donne un exemple dans lequel l'objet “MultiDevice01” comporte trois attributs “FaxBoudRate”, “PrintSpeed” et “FaxSpeed” et deux interfaces “PowerSupply” et “USB”. L'objet “MultiDevice01” prend en charge trois rôles “Printer”, “Fax” et “Scanner”. Le rôle référencé avec la balise “RefBaseRoleClassPath” de l'élément RoleRequirements correspondant représente éventuellement le rôle principal. La Figure A.38 présente le code XML correspondant.

Il convient de mettre en correspondance les attributs et les interfaces appartenant à l'objet “MultiDevice01” avec les attributs et les interfaces de l'ensemble des trois rôles associés. Cette mise en correspondance s’effectue au moyen du MappingObject CAEX conformément au A.2.10 de la CEI 62424:2008, qui fournit des informations précisant les attributs/interfaces de rôle qui sont associés aux attributs/interfaces d'instance. Afin de différencier les attributs des rôles multiples (qui peuvent avoir le même nom), il convient d'inclure le nom de rôle dans la définition du mapping – sauf le rôle principal spécifié par “RefBaseRoleClassPath”. La Figure A.37 présente un exemple correspondant de la méthode de spécification des attributs et interfaces requis, ainsi que de la méthode de mise en correspondance de ces derniers avec les attributs et interfaces d'instance.

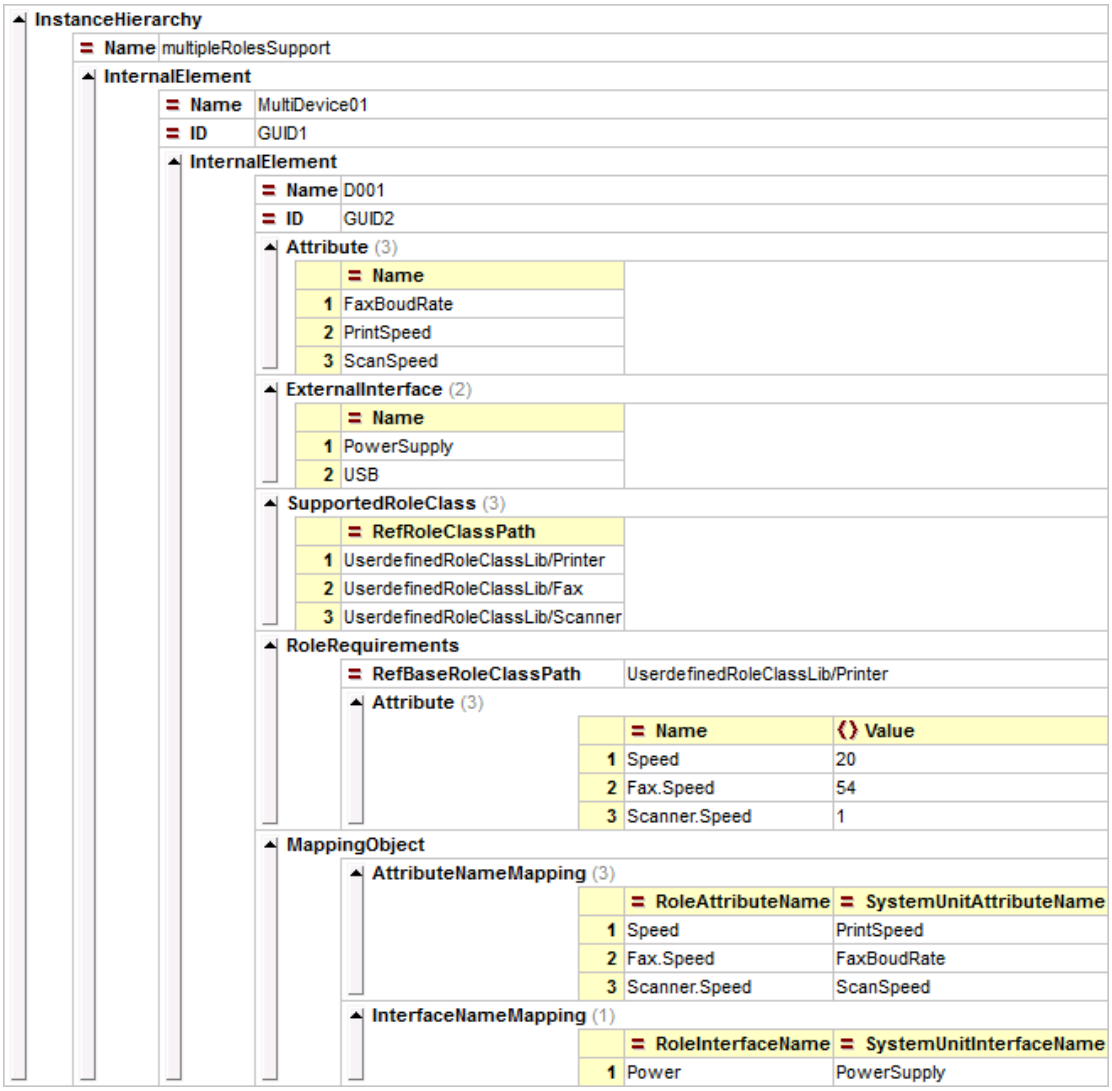


Figure A.37 – Exemple d'une instance définie par l'utilisateur prenant en charge des rôles multiples

La Figure A.38 montre la représentation AML de cette structure.

```
<InstanceHierarchy Name="multipleRolesSupport">
  <InternalElement Name="MultiDevice01" ID="GUID1">
    <InternalElement Name="D001" ID="GUID2">
      <Attribute Name="FaxBoudRate"/>
      <Attribute Name="PrintSpeed"/>
      <Attribute Name="ScanSpeed"/>
      <ExternalInterface Name="PowerSupply"/>
      <ExternalInterface Name="USB"/>
      <SupportedRoleClass RefRoleClassPath="UserdefinedRoleClassLib/Printer"/>
      <SupportedRoleClass RefRoleClassPath="UserdefinedRoleClassLib/Fax"/>
      <SupportedRoleClass RefRoleClassPath="UserdefinedRoleClassLib/Scanner"/>
      <RoleRequirements RefBaseRoleClassPath="UserdefinedRoleClassLib/Printer">
        <Attribute Name="Speed">
          <Value>20</Value>
        </Attribute>
        <Attribute Name="Fax.Speed">
          <Value>54</Value>
        </Attribute>
        <Attribute Name="Scanner.Speed">
          <Value>1</Value>
        </Attribute>
      </RoleRequirements>
      <MappingObject>
        <AttributeNameMapping RoleAttributeName="Speed" SystemUnitAttributeName="PrintSpeed"/>
        <AttributeNameMapping RoleAttributeName="Fax.Speed" SystemUnitAttributeName="FaxBoudRate"/>
        <AttributeNameMapping RoleAttributeName="Scanner.Speed" SystemUnitAttributeName="ScanSpeed"/>
        <InterfaceNameMapping RoleInterfaceName="Power" SystemUnitInterfaceName="PowerSupply"/>
      </MappingObject>
    </InternalElement>
  </InternalElement>
```

Figure A.38 – Texte XML de la représentation AML de la prise en charge de rôles multiples

La Figure A.39 et la Figure A.40 présentent la bibliothèque de classes de rôles AML correspondante, ainsi que sa représentation XML.

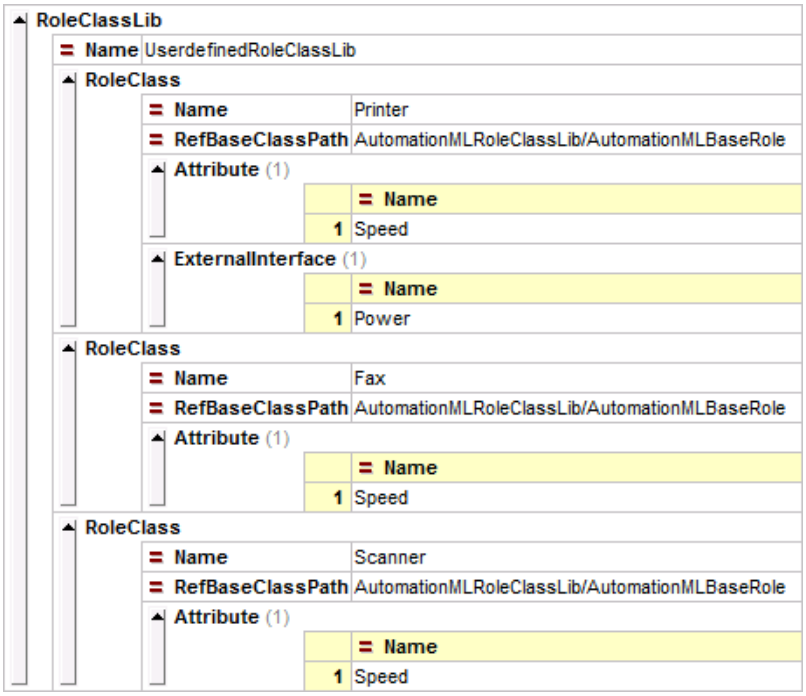


Figure A.39 – Bibliothèque de classes de rôles AML correspondant à l'exemple de définition des rôles multiples

```
<RoleClassLib Name="UserdefinedRoleClassLib">
  <RoleClass Name="Printer" RefBaseClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole">
    <Attribute Name="Speed"/>
    <ExternalInterface Name="Power"/>
  </RoleClass>
  <RoleClass Name="Fax" RefBaseClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole">
    <Attribute Name="Speed"/>
  </RoleClass>
  <RoleClass Name="Scanner" RefBaseClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole">
    <Attribute Name="Speed"/>
  </RoleClass>
</RoleClassLib>
```

Figure A.40 – Texte XML de la bibliothèque de classes de rôles AML

Annexe B
(informative)

Représentation XML des bibliothèques AML

B.1 AutomationMLBaseRoleClassLib

```
<?xml version="1.0" encoding="utf-8"?>
<CAEXFile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd"
   FileName="AutomationMLBaseRoleClassLib.aml" SchemaVersion="2.15">
 <AdditionalInformation AutomationMLVersion="2.0" />
 <AdditionalInformation>
  <WriterHeader>
   <WriterName>IEC SC65E WG 9</WriterName>
   <WriterID>IEC SC65E WG 9</WriterID>
   <WriterVendor>IEC</WriterVendor>
   <WriterVendorURL>www.iec.ch</WriterVendorURL>
   <WriterVersion>1.0</WriterVersion>
   <WriterRelease>1.0.0</WriterRelease>
   <LastWritingDateTime>2013-03-01</LastWritingDateTime>
   <WriterProjectTitle>Automation Markup Language Standard
     Libraries</WriterProjectTitle>
   <WriterProjectID>Automation Markup Language Standard
     Libraries</WriterProjectID>
  </WriterHeader>
 </AdditionalInformation>
 <ExternalReference Path="../InterfaceClass
   Libraries/AutomationMLInterfaceClassLib.aml"
   Alias="AutomationMLInterfaceClassLib" />
 <RoleClassLib Name="AutomationMLBaseRoleClassLib">
  <Description>Automation Markup Language base role class
    library</Description>
  <Version>2.2.0</Version>
  <RoleClass Name="AutomationMLBaseRole">
   <RoleClass Name="Group" RefBaseClassPath="AutomationMLBaseRole">
    <Attribute Name="AssociatedFacet" AttributeDataType="xs:string" />
   </RoleClass>
   <RoleClass Name="Facet" RefBaseClassPath="AutomationMLBaseRole" />
   <RoleClass Name="Port" RefBaseClassPath="AutomationMLBaseRole">
    <Attribute Name="Direction" AttributeDataType="xs:string" />
    <Attribute Name="Cardinality">
     <Attribute Name="MinOccur" AttributeDataType="xs:unsignedInt" />
     <Attribute Name="MaxOccur" AttributeDataType="xs:unsignedInt" />
    </Attribute>
    <Attribute Name="Category" AttributeDataType="xs:string" />
    <ExternalInterface Name="ConnectionPoint" ID="9942bd9c-c19d-44e4-a197-
      11b9edf264e7"
      RefBaseClassPath="AutomationMLInterfaceClassLib@AutomationMLInterfaceC
      lassLib/AutomationMLBaseInterface/PortConnector" />
   </RoleClass>
   <RoleClass Name="Resource" RefBaseClassPath="AutomationMLBaseRole" />
   <RoleClass Name="Product" RefBaseClassPath="AutomationMLBaseRole" />
   <RoleClass Name="Process" RefBaseClassPath="AutomationMLBaseRole" />
   <RoleClass Name="Structure" RefBaseClassPath="AutomationMLBaseRole">
    <RoleClass Name="ProductStructure" RefBaseClassPath="Structure" />
    <RoleClass Name="ProcessStructure" RefBaseClassPath="Structure" />
    <RoleClass Name="ResourceStructure" RefBaseClassPath="Structure" />
   </RoleClass>
   <RoleClass Name="PropertySet" RefBaseClassPath="AutomationMLBaseRole" />
  </RoleClass>
 </RoleClassLib>
</CAEXFile>
```

B.2 AutomationMLInterfaceClassLib

```
<?xml version="1.0" encoding="utf-8"?>
<CAEXFile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd"
FileName="AutomationMLInterfaceClassLib.aml" SchemaVersion="2.15">
 <AdditionalInformation AutomationMLVersion="2.0" />
 <AdditionalInformation>
  <WriterHeader>
   <WriterName>IEC SC65E WG 9</WriterName>
   <WriterID>IEC SC65E WG 9</WriterID>
   <WriterVendor>IEC</WriterVendor>
   <WriterVendorURL>www.iec.ch</WriterVendorURL>
   <WriterVersion>1.0</WriterVersion>
   <WriterRelease>1.0.0</WriterRelease>
   <LastWritingDateTime>2013-03-01</LastWritingDateTime>
   <WriterProjectTitle>Automation Markup Language Standard
    Libraries</WriterProjectTitle>
   <WriterProjectID>Automation Markup Language Standard
    Libraries</WriterProjectID>
  </WriterHeader>
 </AdditionalInformation>
 <InterfaceClassLib Name="AutomationMLInterfaceClassLib">
  <Description>Standard Automation Markup Language Interface Class
   Library</Description>
  <Version>2.2.0</Version>
  <InterfaceClass Name="AutomationMLBaseInterface">
   <InterfaceClass Name="Order" RefBaseClassPath="AutomationMLBaseInterface">
    <Attribute Name="Direction" AttributeDataType="xs:string" />
   </InterfaceClass>
   <InterfaceClass Name="PortConnector"
    RefBaseClassPath="AutomationMLBaseInterface" />
   <InterfaceClass Name="InterlockingConnector"
    RefBaseClassPath="AutomationMLBaseInterface" />
   <InterfaceClass Name="PPRConnector"
    RefBaseClassPath="AutomationMLBaseInterface" />
   <InterfaceClass Name="ExternalDataConnector"
    RefBaseClassPath="AutomationMLBaseInterface">
    <Attribute Name="refURI" AttributeDataType="xs:anyURI" />
    <InterfaceClass Name="COLLADAInterface"
    RefBaseClassPath="ExternalDataConnector" />
    <InterfaceClass Name="PLCopenXMLInterface"
    RefBaseClassPath="ExternalDataConnector" />
   </InterfaceClass>
   <InterfaceClass Name="Communication"
    RefBaseClassPath="AutomationMLBaseInterface">
    <InterfaceClass Name="SignalInterface" RefBaseClassPath="Communication"
      />
    </InterfaceClass>
   </InterfaceClass>
 </InterfaceClassLib>
</CAEXFile>
```

Bibliographie

CEI 60027 (toutes les parties), *Symboles littéraux à utiliser en électrotechnique*

CEI 62264-1, *Intégration des systèmes entreprise-contrôle – Partie 1: Modèles et terminologie*

CEI 62714-2, *Format d'échange de données techniques pour une utilisation dans l'ingénierie des systèmes d'automatisation industrielle – Automation Markup Language – Partie 2: Bibliothèques de classe de rôles* [2]

ISO 80000-1, *Grandeurs et unités – Partie 1: Généralités*

_____

2 A paraître.

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

3, rue de Varembé
PO Box 131
CH-1211 Geneva 20
Switzerland

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