IEC 62714-1

Edition 2.0 2018-04
REDLINE VERSION

INTERNATIONAL STANDARD

colour inside

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

IEC 62714-2018-04 RLV(en)

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

Tel.: +41 22 919 02 11
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 - webstore.iec.ch/advsearchform
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 It also gives information on projects, replaced and withdrawn publications.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 21 000 terms and definitions in English and French, with equivalent terms in 16 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC Glossary - std.iec.ch/glossary
67 000 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: sales@iec.ch.

IEC 62714-1

Edition 2.0 2018-04
REDLINE VERSION

INTERNATIONAL STANDARD

colour inside

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

INTERNATIONAL ELECTROTECHNICAL COMMISSION

ICS 25.040.40; 35.060; 35.240.50

ISBN 978-2-8322-5662-6

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

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.

This redline version of the official IEC Standard allows the user to identify the changes made to the previous edition. A vertical bar appears in the margin wherever a change has been made. Additions are in green text, deletions are in strikethrough red text.

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.

This second edition cancels and replaces the first edition published in 2014. This edition constitutes a technical revision.

This edition includes the following significant technical changes with respect to the previous edition:

a) use of CAEX 3.0 according to IEC 62424:2016 which provides technical improvements as attribute libraries, nested interfaces, new fields for indicating the source of an object, a refinement of the mirror concept and native support of multiple roles, native meta information about the CAEX file source tool, identification of instances via unique IDs instead of pathes, etc.,
b) improved modelling of references to documents outside of the scope of the present standard,
c) modelling of references between CAEX attributes and items in external documents, e.g. within an Excel sheet,
d) revised role libraries,
e) modified Port concept,
f) modelling of multilingual expressions,
g) modelling of structured attribute lists or array,
h) a new AML container format,
i) a new standard AML attribute library.

The text of this International Standard is based on the following documents:

FDIS	Report on voting
65E/582/FDIS	65E/586/RVD

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

This document 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 document will remain unchanged until the stability date indicated on the IEC website under "http://webstore.iec.ch" in the data related to the specific document. At this date, the document 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 for plant engineering data. AML 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. The application of IEC 62714 is industry independent. It is applicable in all industries that require data exchange in their engineering tool chain, e.g. in discrete industry or process industry.

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~~ can 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~~. CAEX is utilized to interconnect 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.

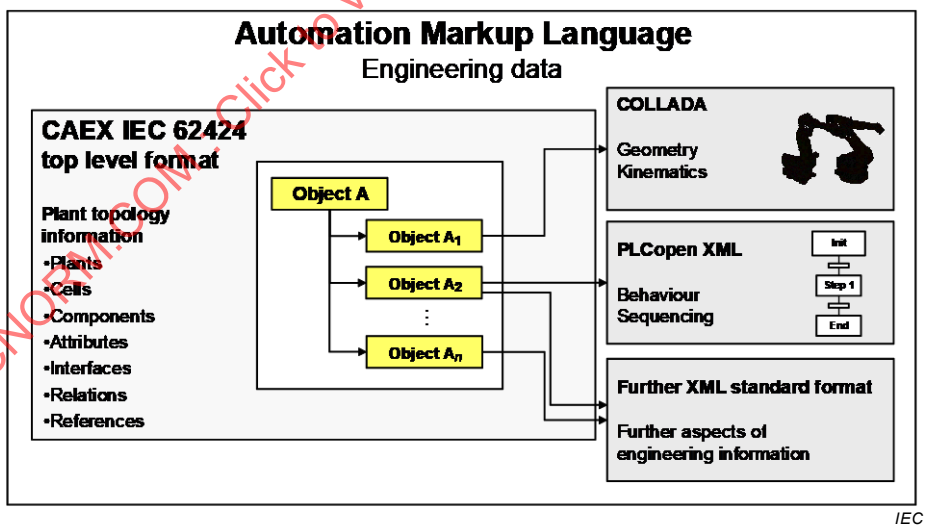


Figure 1 – Overview of the engineering data exchange format AML

Due to the different aspects of AML, the IEC 62714 series consists of different parts focusing 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 subformats.

- IEC 62714-2: Role class libraries

  This part ~~is intended to specify~~ specifies additional AML libraries.

- IEC 62714-3: Geometry and kinematics

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

- IEC 62714-4[1]: Logic

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

Further parts ~~may~~ will 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.

Clause 1 defines the scope for IEC 62714.

Clause 2 provides normative references.

Clause 3 provides terms, definitions and abbreviations.

Clause 4 defines the conformity to IEC 62714.

Clause 5 describes general architecture specifications for IEC 62714.

Clause 6 defines the basic AML libraries.

Clause 7 describes how to model user-defined data.

Clause 8 describes extended AML concepts.

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.

[1] Under consideration.

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 automation markup language (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 are referred to in the text in such a way that some or all of their content constitutes requirements of this document. 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~~ 2016, *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*

ISO/IEC 29500-2, *Information technology – Document description and processing languages – Office Open XML File Formats – Part 2: Open Packaging Conventions*

IETF RFC 2046, *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types* [viewed 2017-11-13]. Available at <http://www.ietf.org>

IETF RFC 4122, A Universally Unique Identifier (UUID) URN Namespace [viewed 2017-11-13]. Available at <http://www.ietf.org>

IETF RFC 5646, *Tags for Identifying Languages* [viewed 2017-11-13]. Available at <http://www.ietf.org>

COLLADA 1.4.1:March 2008, COLLADA – Digital Asset Schema Release 1.4.1 [viewed 2017-11-13]. 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* [viewed 2017-11-13]. 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.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at http://www.electropedia.org/
- ISO Online browsing platform: available at http://www.iso.org/obp

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~~ one or more CAEX RoleRequirements that relate 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, either an AML system unit class, AML interface class, AML role class or AML attribute type

Note 1 to entry: AML classes are stored within AML libraries, AML classes are of type SystemUnitClass, InterfaceClass, RoleClass or AttributeType.

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.

Note 4 to entry: AML classes can be user-defined or standard AML classes.

3.1.5
AML attribute
~~property which belongs to an AML object~~
CAEX attribute which belongs to an AML object and is related to an attribute defined in an AML class or AML AttributeType

Note 1 to entry: AML attributes are described as an XML element corresponding to IEC 62424:~~2008~~ 2016, A.2.4.

3.1.6
AML document
certain AML CAEX document following IEC 62714 (all parts) 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.

Note 2 to entry: AML documents contain AML objects and/or user-defined objects.

Note 3 to entry: An AML document may consist of multiple files, with one AML CAEX document as root.

3.1.7
AML file
certain AML 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~~ with a relation to an AML interface class

Note 1 to entry: Interfaces allow the description of relations between objects by the definition of CAEX Internal-Links. 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~~
AML interface with a direct or indirect relation to the standard AML interface class Port, allowing to specify nested interfaces

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 with a direct or indirect relation to the standard AML role class Group, providing a certain view on AML objects

3.1.12
AML Facet
AML object with a direct or indirect relation to the standard AML role class Facet, 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~~ 2016, 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 automatically 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 (all parts), 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~~

3.1.19
topology
hierarchical structure of a system, visualizable as object tree

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

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

3.1.21
publish
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.

3.1.22
relation
association between CAEX objects

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

3.1.23
link
connection between objects of type CAEX ExternalInterface

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

3.1.24
reference
association between a CAEX InternalElement and externally stored information

3.2 Abbreviations

The abbreviations used in this document are listed in Table 1.

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~~ 2016, 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 focusing 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~~ 2016, 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~~ IEC 62424:2016, A.2.8.7. ~~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~~ IEC 62424:2016, A.2.8.7, 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 (all parts) 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~~ 2016, Annex D.

Reference and relation information: References and relations shall be stored according to 5.5 and 5.6. 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.1.7. 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™[2]. COLLADA interfaces that need to be interconnected within the top level format shall be published as CAEX ExternalInterfaces.

NOTE 5 IEC 62714 (all parts) 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.

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 ~~AM~~ Sub document versions and AML superior document information

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

2 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.

IEC 62714 is based on the following document formats:

- CAEX version ~~2.15~~ 3.0 as defined in IEC 62424:2016;
- 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.

AML integrates CAEX, hence AML acts as superior standard.

NOTE 1 Normative provisions regarding the version information related to AML object instances are defined in 8.7. 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.~~
- Each AML CAEX document shall store the AML version which this standard follows in the CAEX element "SuperiorStandardVersion" according to IEC 62424:2016, A.2.2.3.
- The value of this element shall be "AutomationML 2.10" in order to correspond to the present 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~~ 2.10.

```
<CAEXFile xmlns="http://www.dke.de/CAEX" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
SchemaVersion="3.0" FileName="AutomationML2.10BaseLibraries" xsi:schemaLocation="
http://www.dke.de/CAEX CAEX_ClassModel_V.3.0.xsd">
  <SuperiorStandardVersion>AutomationML 2.10</SuperiorStandardVersion>
```

IEC

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.

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:

- According to IEC 62424:2016, each AML document shall provide information about the source 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~~ can hinder the iterative data exchange with the other tools: hence the removal of writer information of other tools is not recommended.
- This document recommends to use the optional CAEX element SourceObjectInformation with its attributes OriginID and SourceObjID according to IEC 62424:2016, A.2.2.7, in order to identify the source tool of each AML Object instance (InternalElement, ExternalInterface).

~~- 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~~ of the required document origin information. The example shows the source information of the standard libraries provided with this part of IEC 62714.

```
<SourceDocumentInformation OriginID="IEC SC65E WG 9" OriginName="IEC SC65E WG 9"
OriginVersion="2.10.0" LastWritingDateTime="2016-08-25T09:58:00.0Z" OriginProjectID="Automation
Markup Language Standard Library" OriginRelease="2.10.0" OriginVendor="IEC" OriginVendorURL="
www.iec.ch" OriginProjectTitle="Automation Markup Language Standard Libraries"/>
```

IEC

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.~~

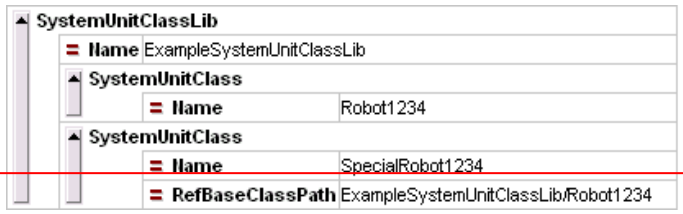
  ~~EXAMPLE 1 "GUID:out".~~

- ~~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".~~

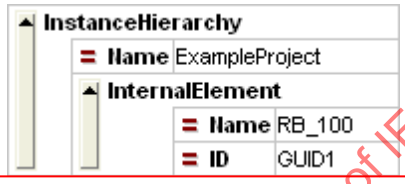
~~Figure 4 gives an example of a SystemUnitClassLib with the SystemUnitClass "Robot1234" and another SystemUnitClass "SpecialRobot1234" derived from "Robot1234".~~




```
<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

5.5 AML relations specification

5.5.1 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 5.5 focuses on relations, whereas 5.6 focuses on references. An informative overview about relations and references is provided in A.1.7.

5.6.2 Parent-child-relations between AML objects

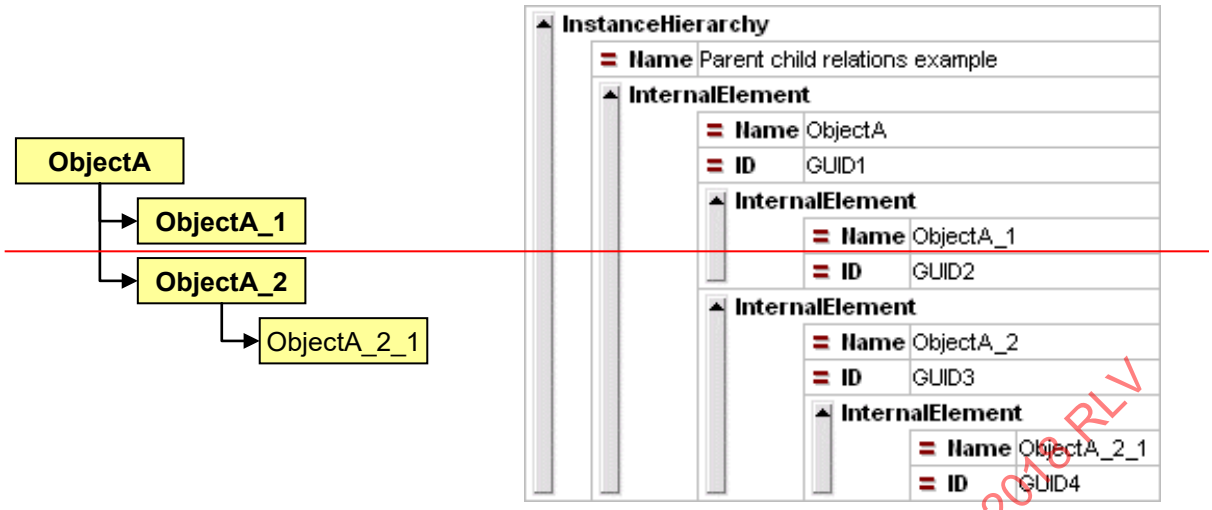
Parent-child-relations between object instances are used to represent hierarchical object structures and describe a "consist-of-relation".

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.

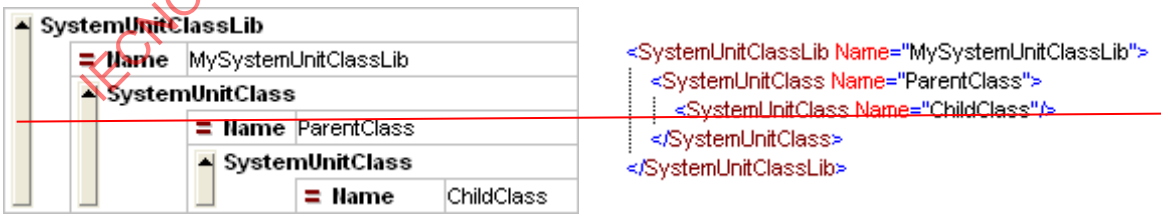


Figure 7 – Example of a parent-child-relation between classes

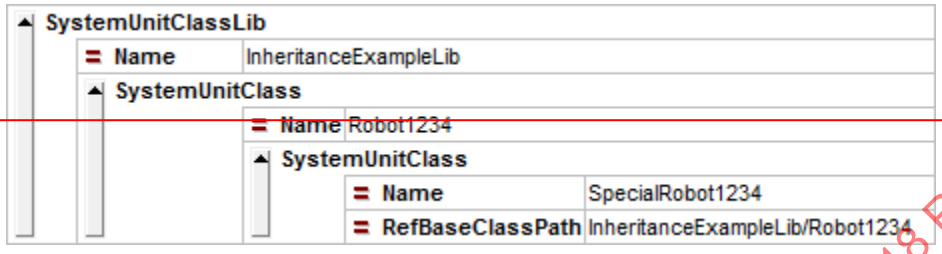
5.6.4 Inheritance relations

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".

5.5.2 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 a CAEX InstanceHierarchy or of a CAEX 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.
- According to IEC 62424:2016, A.2.2.7, 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". Additionally, within the old class, the path to the new version should be stored in the CAEX tag "NewVersion".

  NOTE 3 This provision supports tracking of changes across different versions of a class.
- 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.

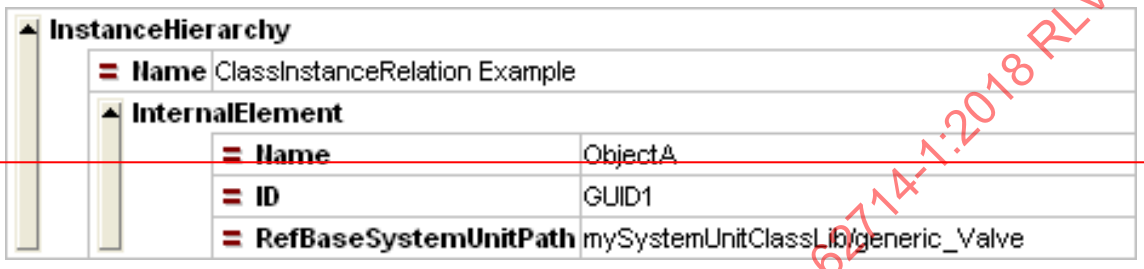
  NOTE 4 The source class is intended to be a suitable starting point for the instance model.
- The copied source class shall be indicated in the CAEX tag "RefBaseSystemUnitPath" of the instance for further usage. This tag shall comprise the full path and name of the source class.

  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.
- 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.

5.5.3 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~~ an AML standard interface class~~es~~.

  NOTE 1 The AML standard interface class library is specified in 6.3. The AML interface class~~es~~ define~~s~~ 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.

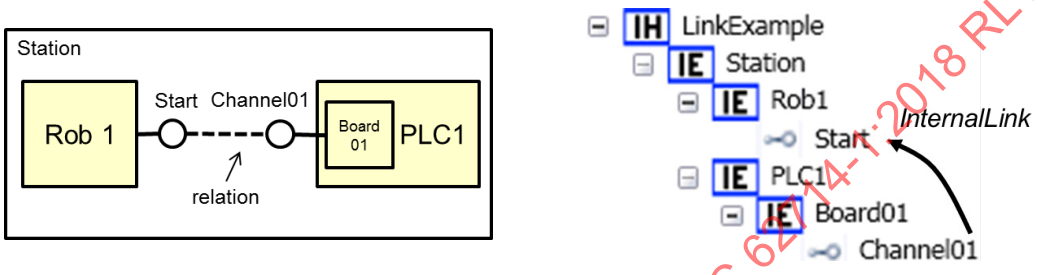
- COLLADA documents may be interlinked. The corresponding COLLADA interfaces may be any items that have a valid URI. If those nodes are required to be interlinked in CAEX, they shall be published in CAEX by adding a CAEX ExternalInterface to the corresponding object. This ExternalInterface shall be derived 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 4a) describes an example comprising a robot “Rob1” and a PLC “PLC1”, each with one signal interface that are connected. Figure 4b) depicts this example as an object hierarchy.



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

Figure 4 – Example of a relation as block diagram and as object tree

Figure 5 and Figure 6 depict the AML representation of the given example. The full XML text for the InstanceHierarchy for this example comprising all ~~AML objects~~ InternalElements “Station”, “Rob1”, “PLC1” and “Board01” including their interfaces is shown below.

NOTE 4 The path strings given in this example (see Figure 4) are reduced with “/.../” in order to increase the readability.

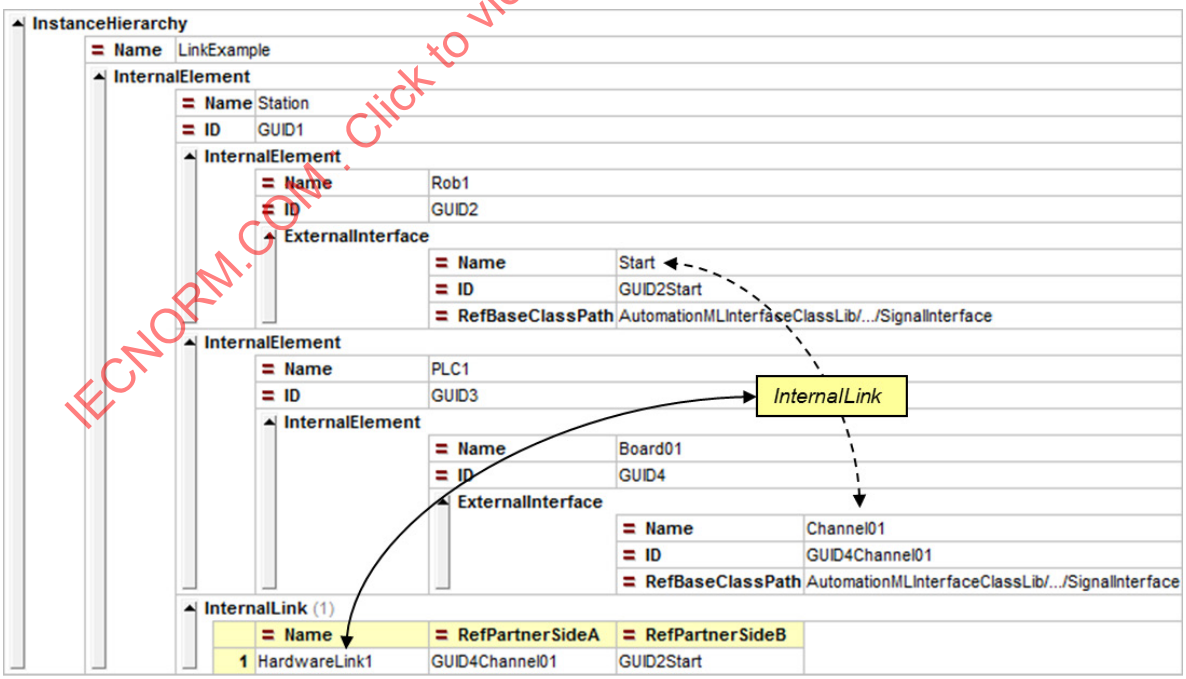


Figure 5 – Example relation between the objects “PLC1” and “Rob1”

```
<InstanceHierarchy Name="LinkExample">
  <InternalElement Name="Station" ID="GUID1">
    <InternalElement Name="Rob1" ID="GUID2">
      <ExternalInterface Name="Start" ID="GUID2Start" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface"/>
    </InternalElement>
    <InternalElement Name="PLC1" ID="GUID3">
      <InternalElement Name="Board01" ID="GUID4">
        <ExternalInterface Name="Channel01" ID="GUID4Channel01" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface"/>
      </InternalElement>
    </InternalElement>
    <InternalLink Name="HardwareLink1" RefPartnerSideA="GUID4Channel01" RefPartnerSideB="GUID2Start"/>
  </InternalElement>
</InstanceHierarchy>
```

IEC

Figure 6 – XML text of the example relation between the objects "PLC1" and "Rob1"

5.5.4 Identification of objects

This document recommends the identification of InternalElements and ExternalInterfaces by means of GUIDs according to RFC 4122. For the comparison of two identifiers, the following provision applies:

- If the identifier of an InternalElement or an ExternalInterface is a GUID, then two identifiers shall be identical, when the contained numerical value is identical. Brackets, braces, dashes or spaces are allowed but shall be irrelevant.

  Example without braces: 48d23207-09e0-4104-82fb-344007d2b7f5

  Example with braces: { 48d23207-09e0-4104-82fb-344007d2b7f5 }

5.6 AML document reference specification

5.6.1 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.

5.6.2 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.

5.6.3 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.

5.6.4 Referencing additional documents in the scope of IEC 62714 (all parts)

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 part of the series. 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.~~

5.6.5 Referencing documents outside of the scope of IEC 62714 (all parts)

If an external document, which is not in the scope of this standard, needs to be referenced by the AML document (e.g. manuals, instructions or specific engineering results like native control programs) the following provisions apply:

- A document which is outside of the scope of IEC 62714 shall be modelled by means of a CAEX InternalElement with a direct or indirect association to the RoleClass "ExternalData" which is defined in 6.4.12. The referenced RoleClass shall specify the content of the document. More than one role with content types can be assigned to a document.

  NOTE 1 Each document can contain contents of several types, e.g. bill of material and user manual.

  NOTE 2 Each document can reference to several files if necessary, e.g. if it splitted in different files.

- If a document is language specific, it shall contain a CAEX attribute of type "DocLang". If a document contains more than one language, it shall contain an unsorted attribute list as described in 8.8 with attributes of type "DocLang".
- Each document shall contain one or more ExternalInterfaces which shall be directly or indirectly derived from the interface class "ExternalDataReference".
- This ExternalInterface shall model the URI to the external document by means of the predefined CAEX attribute of type "refURI" which is inherited from the AML standard interface class "ExternalDataConnector", and shall additionally model the type of the document by means of the predefined CAEX attribute "MIMEType" of type "MIMEType" which is inherited from the AML standard interface class "ExternalDataReference".

Additional information and an example are provided in A.1.5.

5.6.6 Referencing CAEX attributes to items in external documents

If a CAEX attribute needs to be associated with a related item in an external document (e.g. an external XML document or an Excel document which is outside of the scope of this standard), the following provisions apply:

- Each reference between a CAEX attribute and an item in an external document shall be modelled by a CAEX ExternalInterface according to the provisions specified in 5.6.5.
- For each reference between a CAEX attribute and an item in an external document, this CAEX ExternalInterface models one additional attribute of type "AssociatedExternalValue" which has nested attributes.
- The first nested attribute shall mirror the CAEX attribute. This means, that the GUID of the attributes parent object, separated by "/" and the name of the attribute is modelled in the CAEX attribute "RefAttributeType". The name of this attribute shall be irrelevant, but unique among its siblings.
- The second nested attribute of type "refURI" shall reference the item in the external document. This reference shall be the same document or a sub-document of the external document that is referenced in the parent InternalElement defined as described in 5.6.5. The syntax of this reference is outside of the scope of this standard and requires a referencable external document element.
- The third nested attribute of type "Direction" shall model the direction of the information flow. The attribute value shall be "In" if the external attribute is consumed by the CAEX attribute (then the external item is leading), or the attribute value shall be "Out" if the CAEX attribute is leading and the external item consumes the value of the CAEX attribute. The value "InOut" is forbidden.

Additional information and an example are provided in A.1.6.

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~~ after the instantiation 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 standard role class "AutomationMLBaseRole".
- All AML interfaces shall be associated directly or indirectly to ~~an AML~~ the standard interface class "AutomationMLBaseInterface".
- ~~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~~ 2016, Clause 7, Annex A and Annex C. IEC 62714 (all parts) utilizes the CAEX interface concept. User-defined extensions of this AML library are allowed as specified in 7.4.

Each interface shall be derived directly or indirectly from a class of the following standard "AutomationMLInterfaceClassLib" according to Table 2. Subclauses 6.3.2 to 6.3.11 specify the interface classes in detail.

Table 2 – Interface classes of the AutomationMLInterfaceClassLib

AML InterfaceClass library	InterfaceClass	Description
AutomationMLInterfaceClassLib AutomationMLBaseInterface Order Port PPRConnector ExternalDataConnector COLLADAInterface PLCopenXMLInterface ExternalDataReference Communication SignalInterface	AutomationMLBaseInterface	Abstract interface type
	Order	Interface for describing orders
	~~PortConnector~~	~~Interface for describing ports~~
	Port	Interface for describing and interconnecting 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
	ExternalDataReference	Interface to external documents outside of the scope of this standard
	Communication	Generic communication interface
	SignalInterface	Generic signal interface

Figure 7 shows a table view and Figure 8 the XML text of the standard AML InterfaceClassLib. Subclauses 6.3.2 to 6.3.10 provide detail information about the classes.

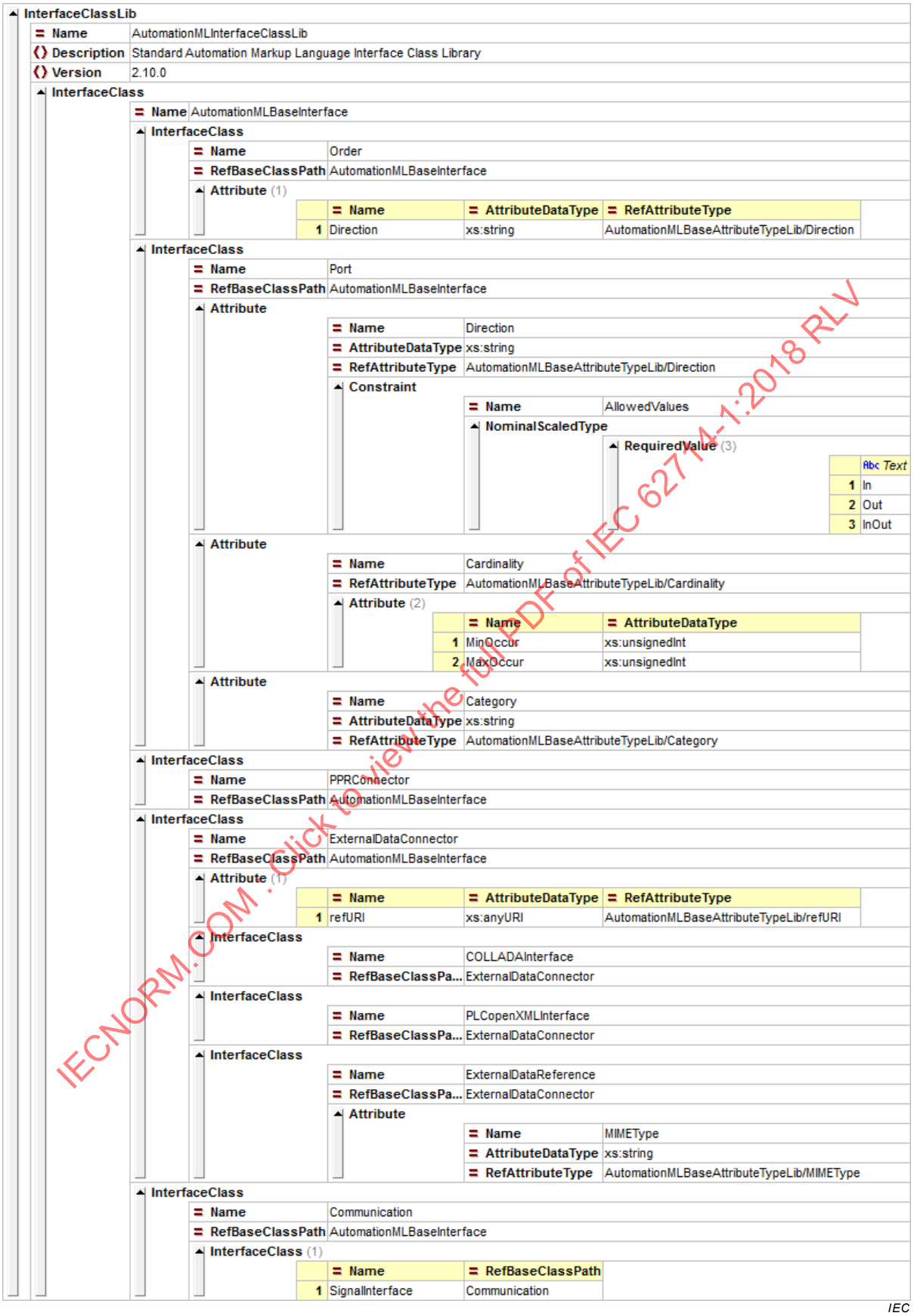


Figure 7 – AML basic interface class library

```
<InterfaceClassLib Name="AutomationMLInterfaceClassLib">
  <Description>Standard Automation Markup Language Interface Class Library</Description>
  <Version>2.10.0</Version>
  <InterfaceClass Name="AutomationMLBaseInterface">
    <InterfaceClass Name="Order" RefBaseClassPath="AutomationMLBaseInterface">
      <Attribute Name="Direction" AttributeDataType="xs:string" RefAttributeType="AutomationMLBaseAttributeTypeLib/Direction"/>
    </InterfaceClass>
    <InterfaceClass Name="Port" RefBaseClassPath="AutomationMLBaseInterface">
      <Attribute Name="Direction" AttributeDataType="xs:string" RefAttributeType="AutomationMLBaseAttributeTypeLib/Direction">
        <Constraint Name="AllowedValues">
          <NominalScaledType>
            <RequiredValue>In</RequiredValue>
            <RequiredValue>Out</RequiredValue>
            <RequiredValue>InOut</RequiredValue>
          </NominalScaledType>
        </Constraint>
      </Attribute>
      <Attribute Name="Cardinality" RefAttributeType="AutomationMLBaseAttributeTypeLib/Cardinality">
        <Attribute Name="MinOccur" AttributeDataType="xs:unsignedInt"/>
        <Attribute Name="MaxOccur" AttributeDataType="xs:unsignedInt"/>
      </Attribute>
      <Attribute Name="Category" AttributeDataType="xs:string" RefAttributeType="AutomationMLBaseAttributeTypeLib/Category"/>
    </InterfaceClass>
    <InterfaceClass Name="PPRConnector" RefBaseClassPath="AutomationMLBaseInterface"/>
    <InterfaceClass Name="ExternalDataConnector" RefBaseClassPath="AutomationMLBaseInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI" RefAttributeType="AutomationMLBaseAttributeTypeLib/refURI"/>
      <InterfaceClass Name="COLLADAInterface" RefBaseClassPath="ExternalDataConnector"/>
      <InterfaceClass Name="PLCopenXMLInterface" RefBaseClassPath="ExternalDataConnector"/>
      <InterfaceClass Name="ExternalDataReference" RefBaseClassPath="ExternalDataConnector">
        <Attribute Name="MIMEType" AttributeDataType="xs:string" RefAttributeType="AutomationMLBaseAttributeTypeLib/MIMEType"/>
      </InterfaceClass>
    </InterfaceClass>
    <InterfaceClass Name="Communication" RefBaseClassPath="AutomationMLBaseInterface">
      <InterfaceClass Name="SignalInterface" RefBaseClassPath="Communication"/>
    </InterfaceClass>
  </InterfaceClass>
</InterfaceClassLib>
```

IEC

Figure 8 – XML description of the AML basic interface class library

6.3.2 InterfaceClass AutomationMLBaseInterface

Table 3 specifies the interface class "AutomationMLBaseInterface".

Table 3 – 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
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface
Attributes	None

6.3.3 InterfaceClass Order

Table 4 specifies the interface class “Order”.

Table 4 – 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
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Order
Attributes	~~Direction (type=”xs:string”)~~ ~~The attribute “Direction” shall be used in order to specify the direction. Permitted values are “In”, “Out” or “InOut”.~~ Name: Direction RefAttributeType: AutomationMLBaseAttributeTypeLib/Direction Semantics: see 6.5.2

~~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~~

6.3.4 InterfaceClass Port

Table 5 specifies the role class “Port”.

Table 5 – Optional attributes for AML Port interfaces

Class name	Port
Description	The interface class “Port” is an interface type for interfaces that contain a number of nested interfaces and allows describing complex interfaces in this way. AML Port interfaces shall reference this interface class. Details and examples are specified in 8.2.
Parent class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Port
Attributes	Name: Direction RefAttributeType: AutomationMLBaseAttributeTypeLib/Direction Semantics: see 6.5.2
	Name: Cardinality RefAttributeType: AutomationMLBaseAttributeTypeLib/Cardinality Semantics: see 6.5.2
	Name: Category RefAttributeType: AutomationMLBaseAttributeTypeLib/Category Semantics: see 6.5.2

6.3.5 InterfaceClass PPRConnector

Table 6 specifies the interface class “PPRConnector”.

Table 6 – 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.2.5 for more information.
Parent class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/PPRConnector
Attributes	~~None~~

6.3.6 InterfaceClass ExternalDataConnector

Table 7 specifies the interface class “ExternalDataConnector”.

Table 7 – 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
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector
Attributes	~~refURI~~ ~~(type=”xs:anyURI”)~~ ~~The attribute “refURI” shall be used in order to store the path to the reference external document.~~ Name: refURI RefAttributeType: AutomationMLBaseAttributeTypeLib/refURI Semantics: see 6.5.2

6.3.7 InterfaceClass COLLADAInterface

Table 8 specifies the interface class “COLLADAInterface”. Details are ~~intended to be~~ specified in IEC 62714-3.

Table 8 – 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
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ ExternalDataConnector/COLLADAInterface
Attributes	None

6.3.8 InterfaceClass PLCopenXMLInterface

Table 9 specifies the interface class “PLCopenXMLInterface”. Details are ~~intended to be~~ specified in IEC 62714-4.

Table 9 – 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	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ ExternalDataConnector/PLCopenXMLInterface
Attributes	None

6.3.9 InterfaceClass ExternalDataReference

Table 10 specifies the interface class “ExternalDataReference”. Details are specified in 5.6.5.

Table 10 – InterfaceClass ExternalDataReference

Class name	ExternalDataReference
Description	The interface class “ExternalDataReference” shall be used in order to reference external documents out of the scope of AML. Details are specified in 5.6.5.
Parent class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ ExternalDataConnector/ExternalDataReference
Attributes	Name: MIMEType RefAttributeType: AutomationMLBaseAttributeTypeLib/MIMEType Semantics: see 6.5.2

6.3.10 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
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication
Attributes	None

6.3.11 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 4.
Parent class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication/SignalInterface
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.

A role can be associated to a SystemUnitClass by means of its CAEX SupportedRoleClass(es). This indicates that the class is able to support the referenced role(s). Multiple SupportedRoleClasses are supported. The MappingObject provides a mapping between role attributes and interfaces and SystemUnitClass attributes and interfaces. Once a SystemUnitClass is instantiated, the related InternalElement holds this information. However, this does not indicate that the instance actually plays all roles within its individual context.

CAEX RoleRequirements model the actual or requested roles of a CAEX InternalElement. Multiple RoleRequirements are supported. The actual role(s) are referenced, and requirements of the individual instance concerning the role are modelled within the RoleRequirement. The MappingObject allows mapping attributes and interfaces between the role class and the InternalElement, if required.

While associating a role class to an AML object, this AML object gets a semantic. Example role classes are a “Resource” or a “Robot”. 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~~ after instantiation 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 9 to Figure 11 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 6.4.12.

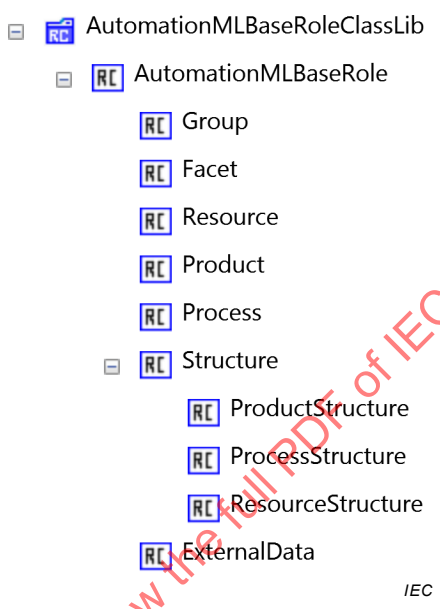


Figure 9 – AML basic role class library

RoleClassLib

Name	AutomationMLBaseRoleClassLib
Description	Automation Markup Language base role class library
Version	2.10.0

RoleClass

Name	AutomationMLBaseRole

RoleClass

Name	Group
RefBaseClassPath	AutomationMLBaseRole

Attribute

Name	AssociatedFacet
AttributeDataType	xs:string
RefAttributeType	AutomationMLBaseAttributeTypeLib/AssociatedFacet

RoleClass

Name	Facet
RefBaseClassPath	AutomationMLBaseRole

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	Structure
2	ProcessStructure	Structure
3	ResourceStructure	Structure

RoleClass

Name	ExternalData
RefBaseClassPath	AutomationMLBaseRole

IEC

Figure 10 – AutomationMLBaseRoleClassLib

```
<RoleClassLib Name="AutomationMLBaseRoleClassLib">
  <Description>Automation Markup Language base role class library</Description>
  <Version>2.10.0</Version>
  <RoleClass Name="AutomationMLBaseRole">
    <RoleClass Name="Group" RefBaseClassPath="AutomationMLBaseRole">
      <Attribute Name="AssociatedFacet" AttributeDataType="xs:string" RefAttributeType="AutomationMLBaseAttributeTypeLib/AssociatedFacet"/>
    </RoleClass>
    <RoleClass Name="Facet" RefBaseClassPath="AutomationMLBaseRole"/>
    <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="ExternalData" RefBaseClassPath="AutomationMLBaseRole"/>
  </RoleClass>
</RoleClassLib>
```

IEC

Figure 11 – 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
Path for element reference	AutomationMLBaseRoleClassLib/AutomationMLBaseRole
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
Path for element reference	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Group
Attributes	~~“AssociatedFacet”~~ ~~(type = “xs:string”)~~ ~~The attribute “AssociatedFacet” shall be used for the definition of the name of the corresponding Facet.~~ ~~Example: AssociatedFacet = “PLCFacet”.~~ Name: AssociatedFacet RefAttributeType: AutomationMLBaseAttributeTypeLib/AssociatedFacet Semantics: see 6.5.2

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
Path for element reference	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Facet
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.~~

6.4.5 RoleClass Resource

Table 16 specifies the role class "Resource".

Table 16 – 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.2.5.
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole
Path for element reference	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource
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).

6.4.6 RoleClass Product

Table 17 specifies the role class “Product”.

Table 17 – 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.2.5.
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole
Path for element reference	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Product
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).

6.4.7 RoleClass Process

Table 18 specifies the role class “Process”.

Table 18 – 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.2.5.
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole
Path for element reference	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Process
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).

6.4.8 RoleClass Structure

Table 19 specifies the role class “Structure”.

Table 19 – 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
Path for element reference	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure
Attributes	None

6.4.9 RoleClass ProductStructure

Table 20 specifies the role class “ProductStructure”.

Table 20 – 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
Path for element reference	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure/ ProductStructure
Attributes	None

6.4.10 RoleClass ProcessStructure

Table 21 specifies the role class “ProcessStructure”.

Table 21 – 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
Path for element reference	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure/ ProcessStructure
Attributes	None

6.4.11 RoleClass ResourceStructure

Table 22 specifies the role class “ResourceStructure”.

Table 22 – 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
Path for element reference	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure/ ResourceStructure
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~~

6.4.12 RoleClass ExternalData

Table 23 specifies the role class “ExternalData”.

Table 23 – RoleClass ExternalData

Class name	ExternalData
Description	The role class “ExternalData” is an abstract role type for a document type and the base class for all document type roles. It describes different document types. AML document objects shall directly or indirectly reference this role. Examples are specified in A.1.5.
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole
Path for element reference	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/ExternalData
Attributes	None

6.5 AML basic attribute type library

6.5.1 General

Subclause 6.5 defines AML basic attribute types, required by the standard AML classes defined in 6.3 and 6.4. Figure 12 and Figure 13 present the standard basic attribute type library as XML table and as XML text. Details of each attribute type are given in 6.5.2.

AttributeTypeLib

Name	AutomationMLBaseAttributeTypeLib
Description	Standard Automation Markup Language Attribute Type Library
Version	2.10.0

AttributeType

Name	Direction
AttributeDataType	xs:string

Constraint

Name	AllowedValues

NominalScaledType

RequiredValue (3)

	Text
1	In
2	Out
3	InOut

AttributeType

Name	Cardinality

Attribute (2)

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

AttributeType

Name	Category
AttributeDataType	xs:string

AttributeType

Name	refURI
AttributeDataType	xs:anyURI

AttributeType

Name	AssociatedFacet
AttributeDataType	xs:string

AttributeType

Name	ListType

AttributeType

Name	OrderedListType

AttributeType

Name	LocalizedAttribute
AttributeDataType	xs:string

AttributeType

Name	AssociatedExternalValue

Attribute (3)

	Name	RefAttributeType
1	refCAEXAttribute	
2	refURI	AutomationMLBaseAttributeTypeLib/refURI
3	Direction	AutomationMLBaseAttributeTypeLib/Direction

AttributeType

Name	MIMEType
AttributeDataType	xs:string

AttributeType

Name	DocLang
AttributeDataType	xs:string

IEC

Figure 12 – AML basic attribute type library

```
<AttributeTypeLib Name="AutomationMLBaseAttributeTypeLib">
  <Description>Standard Automation Markup Language Attribute Type Library</Description>
  <Version>2.10.0</Version>
  <AttributeType Name="Direction" AttributeDataType="xs:string">
    <Constraint Name="AllowedValues">
      <NominalScaledType>
        <RequiredValue>In</RequiredValue>
        <RequiredValue>Out</RequiredValue>
        <RequiredValue>InOut</RequiredValue>
      </NominalScaledType>
    </Constraint>
  </AttributeType>
  <AttributeType Name="Cardinality">
    <Attribute Name="MinOccur" AttributeDataType="xs:unsignedInt"/>
    <Attribute Name="MaxOccur" AttributeDataType="xs:unsignedInt"/>
  </AttributeType>
  <AttributeType Name="Category" AttributeDataType="xs:string"/>
  <AttributeType Name="refURI" AttributeDataType="xs:anyURI"/>
  <AttributeType Name="AssociatedFacet" AttributeDataType="xs:string"/>
  <AttributeType Name="ListType"/>
  <AttributeType Name="OrderedListType"/>
  <AttributeType Name="LocalizedAttribute" AttributeDataType="xs:string"/>
  <AttributeType Name="AssociatedExternalValue">
    <Attribute Name="refCAEXAttribute"/>
    <Attribute Name="refURI" RefAttributeType="AutomationMLBaseAttributeTypeLib/refURI"/>
    <Attribute Name="Direction" RefAttributeType="AutomationMLBaseAttributeTypeLib/Direction"/>
  </AttributeType>
  <AttributeType Name="MIMEType" AttributeDataType="xs:string"/>
  <AttributeType Name="DocLang" AttributeDataType="xs:string"/>
</AttributeTypeLib>
```

IEC

Figure 13 – XML text of the AutomationMLBaseAttributeTypeLib

6.5.2 Attributes of the AutomationMLBaseAttributeTypeLib

Table 24 specifies the attribute types of the AutomationMLBaseAttributeTypeLib.

Table 24 – Attribute Types of the AutomationMLBaseAttributeTypeLib

Attribute name	Semantics
Direction	This attribute shall be used to describe a direction of a CAEX interface, e.g. of a signal or of an interface. Allowed values: “In”, “Out” or “InOut”. CAEX Interfaces using this attribute follow the following provisions: • Interfaces with the direction “In” shall only be connected to interfaces with the direction “Out” or “InOut”. • Interfaces with the direction “Out” shall only be connected to interfaces with the direction “In” or “InOut”. This information can be used e.g. in order to prove the validity of a connection. Examples: • Direction = “Out” (e.g. a plug) • Direction = “In” (e.g. a socket) • Direction = “InOut” NOTE The validity of those connections is outside the scope of IEC 62714, but is a tool functionality. AttributeDataType: xs:string Path: AutomationMLBaseAttributeTypeLib/Direction

Attribute name	Semantics
Cardinality	This attribute belongs to a CAEX ExternalInterface and shall be used to describe the allowed maximum and minimum numbers of connections from/to this interface. The attribute Cardinality itself is a complex attribute and shall not have a value. The corresponding sub-attributes are described in Table 25. AttributeDataType: This attribute has no AttributeDataType since the attribute has no value. Path: AutomationMLBaseAttributeTypeLib/Cardinality
Category	This attribute belongs to a CAEX ExternalInterface and describes the category of this interface. The value of this attribute is user-defined. Only interface classes with the same category value are allowed to be connected. This standard does not predefine category values. Example: Category = “MaterialFlow”. AttributeDataType: xs:string Path: AutomationMLBaseAttributeTypeLib/Category
refURI	This attribute shall be used in order to store a path to an external document. AttributeDataType: xs:anyURI Path: AutomationMLBaseAttributeTypeLib/refURI
AssociatedFacet	The attribute “AssociatedFacet” shall be used for the definition of the name of a related Facet. The Facet concept is described in 8.3. Example: AssociatedFacet = “PLCFacet”. AttributeDataType: xs:string Path: AutomationMLBaseAttributeTypeLib/AssociatedFacet
ListType	The attribute “ListType” shall be used for attributes that contain an unsorted list of attributes. The concept is described in A.2.7. AttributeDataType: empty Path: AutomationMLBaseAttributeTypeLib/ListType
OrderedListType	The attribute “OrderedListType” shall be used for attributes that contain a sorted list of attributes. The concept is described in A.2.7. AttributeDataType: empty Path: AutomationMLBaseAttributeTypeLib/OrderedListType
LocalizedAttribute	The attribute “LocalizedAttribute” shall be used for sub-attributes describing a language alternative of its parent attribute. The language according to RFC 5646 shall be used as name of the attribute. The concept is described in A.2.6. AttributeDataType: xs:string Path: AutomationMLBaseAttributeTypeLib/LocalizedAttribute
AssociatedValue	The AssociatedValue contains nested attributes which allow to interconnect a CAEX attribute to an item in an external document. The concept is described in A.1.6. The attribute AssociatedValue itself is a complex attribute and shall not have a value. The corresponding sub-attributes are described in Table 26. AttributeDataType: This attribute has no AttributeDataType since the attribute has no value. Parent: AutomationMLBaseAttributeTypeLib Path: AutomationMLBaseAttributeTypeLib/AssociatedValue

Attribute name	Semantics
MIMEType	The MIMEType describes the MIMEType of a referenced document. The attribute shall have values according to RFC 2046. Examples: MIMEType = “application/pdf” – this means that this document is of file type pdf. MIMEType = “application/xml” – this means that this document is of type xml. AttributeDataType: xs:string Parent: AutomationMLBaseAttributeTypeLib Path: AutomationMLBaseAttributeTypeLib/MIMEType
DocLang	The DocLang describes the language of a referenced document. The attribute shall have a value according to RFC 5646. Example: DocLang = “fr-FR”, “en-US”, “de-DE” – This means that this document is in French, American English or German language valid in France, US or Germany. AttributeDataType: xs:string Parent: AutomationMLBaseAttributeTypeLib Path: AutomationMLBaseAttributeTypeLib/DocLang

Table 25 – Sub-attributes of the attribute “Cardinality”

Attribute	AttributeDataType	Description	Example
MinOccur	xs:unsignedInt	The MinOccur value describes the minimum possible number of connections to or from the corresponding interface class. 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 the corresponding interface class. The attribute shall have values greater than or equal to Min-Occur, or 0 which means infinite.	MaxOccur = 3 This means that this Port can only be connected with a maximum of three other ports.

Table 26 – Sub-attributes of the attribute “AssociatedValue”

Attribute	RefAttributeType	Description	Example
refCAEXAttribute	<GUID/attName>	This attribute mirrors the CAEX attribute that should be interlinked with another item in an external document. This attribute has no value.	refCAEXAttribute= GUID1/Temperatur
refURI	AutomationMLBase-AttributeTypeLib/refURI	See refURI in Table 24	
Direction	AutomationMLBase-AttributeTypeLib/Direction	See Direction in Table 24	

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 SystemUnitClasses, CAEX Attributes ~~–~~ Types, CAEX InterfaceClasses and CAEX RoleClasses which are not predefined by IEC 62714 (all parts). 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 (all parts). 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~~ Attribute Library Concept or standard CAEX mappings. These concepts ease the automatic interpretation of user-defined classes, attribute types and attributes.

7.2 User-defined attributes

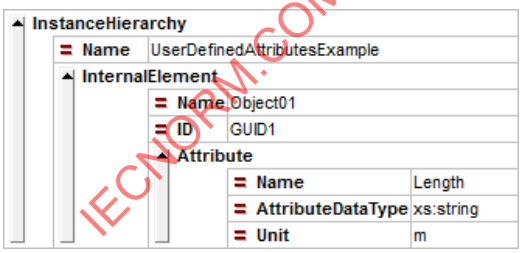
~~All~~ Attributes defined in IEC 62714 (all parts) are AML standard attributes. ~~All~~ Attributes which are not defined in IEC 62714 (all parts) are user-defined attributes. All attributes are stored in the same way as CAEX Attributes.

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~~ 2016, 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 14 gives an example of a user-defined object “Object01” with a user-defined attribute “Length”.




```
<InstanceHierarchy Name="UserDefinedAttributesExample">
  <InternalElement Name="Object01" ID="GUID1">
    <Attribute Name="Length" AttributeDataType="xs:string" Unit="m"/>
  </InternalElement>
</InstanceHierarchy>
```

IEC

Figure 14 – Example of a user-defined attribute

7.3 User-defined AttributeTypes

Regarding user-defined AttributeTypes, the following provisions apply:

- User-defined AttributeTypes shall be stored in AML according to the CAEX AttributeType definition in IEC 62424:2016, A.2.5.
- AttributeTypeLibraries are recommended to model user specific, company specific, region specific, country specific, etc. or normative international standards with respect to attribute semantics, related to todays or future attribute standards. This is the basis for the storage

of agreed attribute semantics and enables independence from language and naming conventions.

EXAMPLE: Figure 15 and Figure 16 illustrate the definition of user-defined AttributeTypes.

- The AttributeTypeLibrary “MyAttributeTypeLibrary” contains the AttributeTypes “Height”, “Width” and “Length”.
- The InternalElement “Station” has 3 attributes referencing the user-defined Attribute types.

NOTE The Attribute Type library concept allows to name object attributes different to the AttributeType. This enables independence from language and naming conventions.

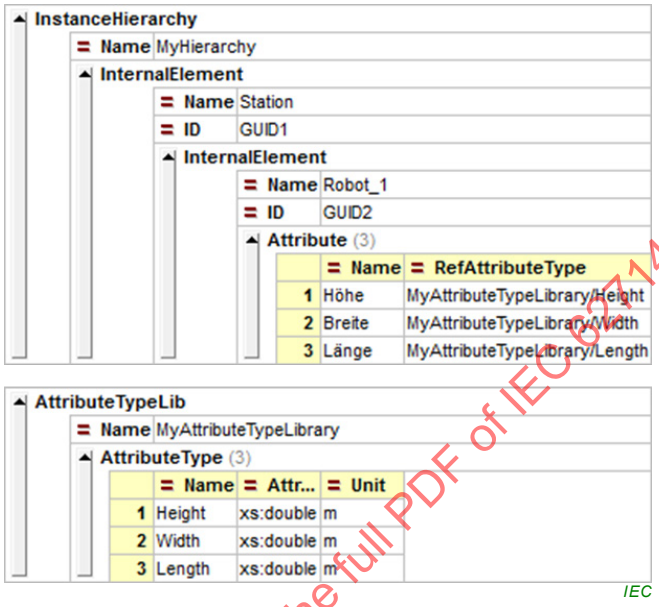


Figure 15 – Examples for user-defined AttributeTypes

```
<InstanceHierarchy Name="MyHierarchy">
  <InternalElement Name="Station" ID="GUID1">
    <InternalElement Name="Robot_1" ID="GUID2">
      <Attribute Name="Höhe" RefAttributeType="MyAttributeTypeLibrary/Height"/>
      <Attribute Name="Breite" RefAttributeType="MyAttributeTypeLibrary/Width"/>
      <Attribute Name="Länge" RefAttributeType="MyAttributeTypeLibrary/Length"/>
    </InternalElement>
  </InternalElement>
```

```
<AttributeTypeLib Name="MyAttributeTypeLibrary">
  <AttributeType Name="Height" AttributeDataType="xs:double" Unit="m"/>
  <AttributeType Name="Width" AttributeDataType="xs:double" Unit="m"/>
  <AttributeType Name="Length" AttributeDataType="xs:double" Unit="m"/>
</AttributeTypeLib>
```

IEC

Figure 16 – XML code of the examples for user-defined AttributeTypes

7.4 User-defined InterfaceClasses

All InterfaceClasses defined in IEC 62714 (all parts) are standard AML interface classes. All InterfaceClasses not defined in IEC 62714 (all parts) are 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~~ IEC 62424:2016, A.2.6.

  NOTE ~~AML InterfaceClasses and User-defined InterfaceClasses~~ User-defined and standard AML interface classes 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 interface classes.

Figure 17 and Figure 18 show 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.

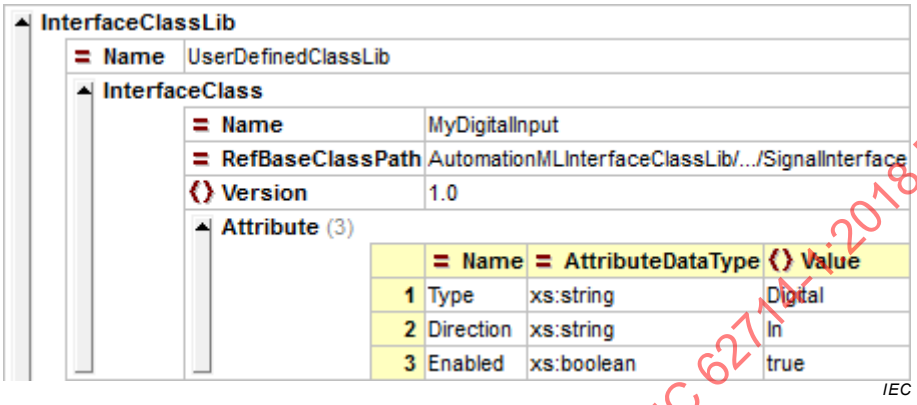


Figure 17 – Example of a user-defined InterfaceClass in a user-defined InterfaceClassLib

```
<InterfaceClassLib Name="UserDefinedClassLib">
  <InterfaceClass Name="MyDigitalInput" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface">
    <Version>1.0</Version>
    <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>
```

IEC

Figure 18 – XML code of the example of a user-defined InterfaceClass in a user-defined InterfaceClassLib

7.5 User-defined RoleClasses

All RoleClasses defined in IEC 62714 (all parts) are standard AML role classes. All role classes not defined in IEC 62714 (all parts) are 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~~ IEC 62424:2016, A.2.7.

  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 and Figure 20 show 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.

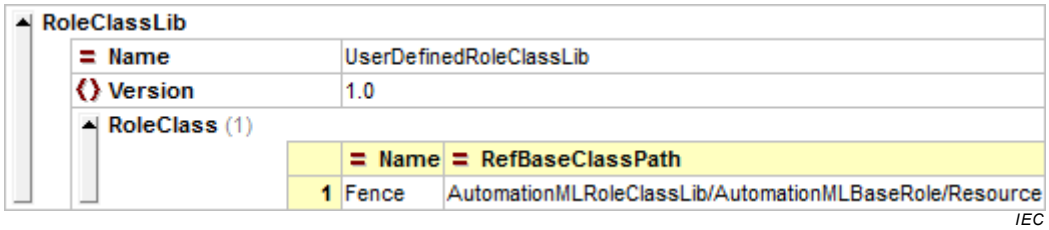


Figure 19 – Example of a user-defined RoleClass in a user-defined RoleClassLib

```
<RoleClassLib Name="UserDefinedRoleClassLib">
  <Version>1.0</Version>
  <RoleClass Name="Fence" RefBaseClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource"/>
</RoleClassLib>
```

IEC

Figure 20 – XML code of the example of a user-defined RoleClass in a user-defined RoleClassLib

7.6 User-defined SystemUnitClasses

IEC 62714 does not specify SystemUnitClasses, hence all SystemUnitClasses are userdefined.

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~~ 2016, A.2.3.
- User-defined SystemUnitClasses ~~shall directly or indirectly~~ should be assigned to a~~n AML~~ role class and shall use AML attributes whenever applicable. Hence, user-defined SystemUnitClasses may be assigned to a standard AML role class, a user-defined role class, or no role class.

It is recommended to assign user-defined SystemUnitClasses to a role class. This serves the automatic interpretability.

Examples: Figure 21 and Figure 22 illustrate 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 automatically be interpreted as “Resource”.
- The SystemUnitClass “SpecialRobot1234” depicts a new user-defined class which is derived from “Robot1234”. This class is therefore also a resource.

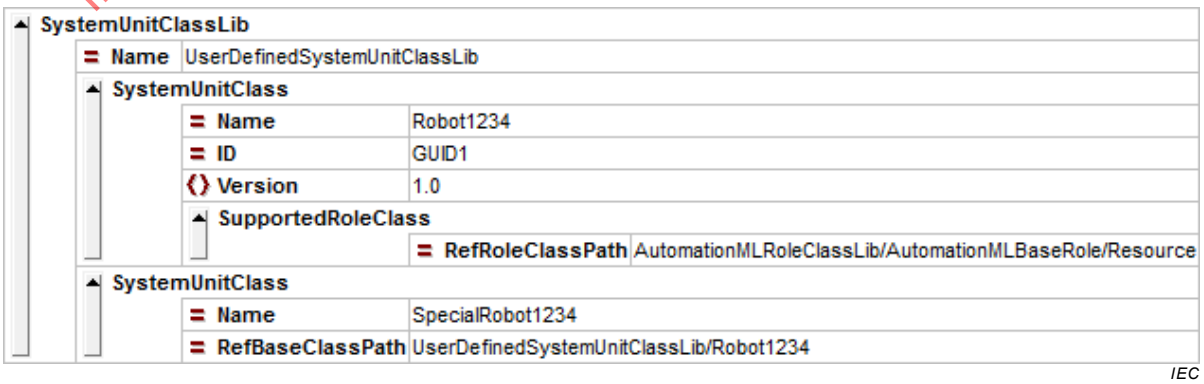


Figure 21 – Examples for different user-defined SystemUnitClasses

```
<SystemUnitClassLib Name="UserDefinedSystemUnitClassLib">
  <SystemUnitClass Name="Robot1234" ID="GUID1">
    <Version>1.0</Version>
    <SupportedRoleClass RefRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource"/>
  </SystemUnitClass>
  <SystemUnitClass Name="SpecialRobot1234" RefBaseClassPath="UserDefinedSystemUnitClassLib/Robot1234"/>
</SystemUnitClassLib>
```

IEC

Figure 22 – XML code of the examples for different user-defined SystemUnitClasses

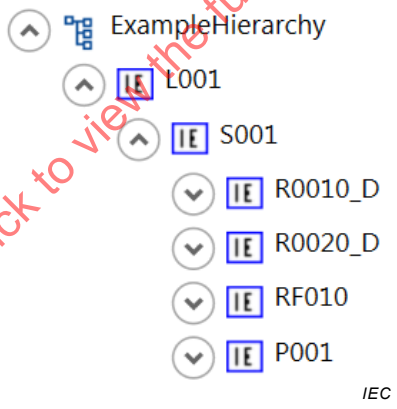
7.7 User-defined InstanceHierarchies

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 23 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”.



IEC

Figure 23 – Example of a user-defined InstanceHierarchy

Figure 24 shows the AML representation of this structure. According to IEC 62424:~~2008~~ 2016, ~~A.2.9~~ A.2.10, 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="ABBRobotLibrary/ABB_Robot_1234">
        <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource/Mechatronic/Robot"/>
      </InternalElement>
      <InternalElement Name="R0020_D" ID="GUID4" RefBaseSystemUnitPath="ABBRobotLibrary/ABB_Robot_1234">
        <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource/Mechatronic/Robot"/>
      </InternalElement>
      <InternalElement Name="RF010" ID="GUID5">
        <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource/Mechatronic/Transport/Rollerbed"/>
      </InternalElement>
      <InternalElement Name="P001" ID="GUID6">
        <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource/ControlEquipment/ControlHardware/PLC"/>
      </InternalElement>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource/Structure/Station"/>
    </InternalElement>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource/Structure/Line"/>
  </InternalElement>
</InstanceHierarchy>
```

IEC

Figure 24 – 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 Clause A.2.

8.2 AML Port ~~object~~ interface

~~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.~~

An AML Port is a CAEX interface allowing the specification of complex (nested) 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 ExternalInterface with a direct or indirect association to the InterfaceClass "Port" which is described in 6.3.4.
- All nested ExternalInterfaces of the Port interface object should directly or indirectly be derived from an AML interface class defined in 6.3.

8.3 AML Facet object

A Facet is an AML object providing a sub-view on attributes or interfaces of the parent ~~AML object~~ CAEX InternalElement. 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 a direct or indirect 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.~~
- An AML Facet object may be modelled at an arbitrary position of the InstanceHierarchy or a SystemUnitClass and shall be a child object of an InternalElement or SystemUnitClass.
- Facets are identified by their unique ID. Their names are display names only.
- An InternalElement or SystemUnitClass may have an arbitrary number of Facet objects.
- An AML Facet object shall only contain mirror attributes or interfaces.
- Facets may have an arbitrary number of facet attributes and facet interfaces.
- Each Facet attribute shall mirror an existing attribute of the parent object, according to IEC 62424:2016, A.2.8.7. Mirroring of sub-attributes and other subordinated attributes within the parent object is possible.
- Facet attributes which are not part of the parent object are not permitted.
- Each Facet interface shall mirror an existing interface of the parent object, according to IEC 62424:2016, A.2.8.7. Mirroring of nested interfaces within the parent object is possible.
- 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 a direct or indirect association to the RoleClass “Group” which is defined in 6.4.3.

- An AML Group object may be modelled at an arbitrary position of a InstanceHierarchy or a SystemUnitClass and shall be a child object of an InstanceHierarchy, InternalElement or 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.~~

- Groups are identified by their unique ID. Their names are display names only.
- An InternalElement or SystemUnitClass may have an arbitrary number of Group objects.
- An AML Group object shall only contain mirror InternalElements and/or further Group objects.

  NOTE 1 Thus, Group objects can be nested.
- AML Groups shall not be used to describe plant hierarchies.
- An AML Group object may store additional information as attributes or interfaces in order to describe group specific information.

  NOTE 2 Those additional attributes or interfaces are not identical to attributes 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.
- If used, the attribute "AssociatedFacet" shall have a value that provides a valid name of an existing Facet of each mirror object.

~~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.

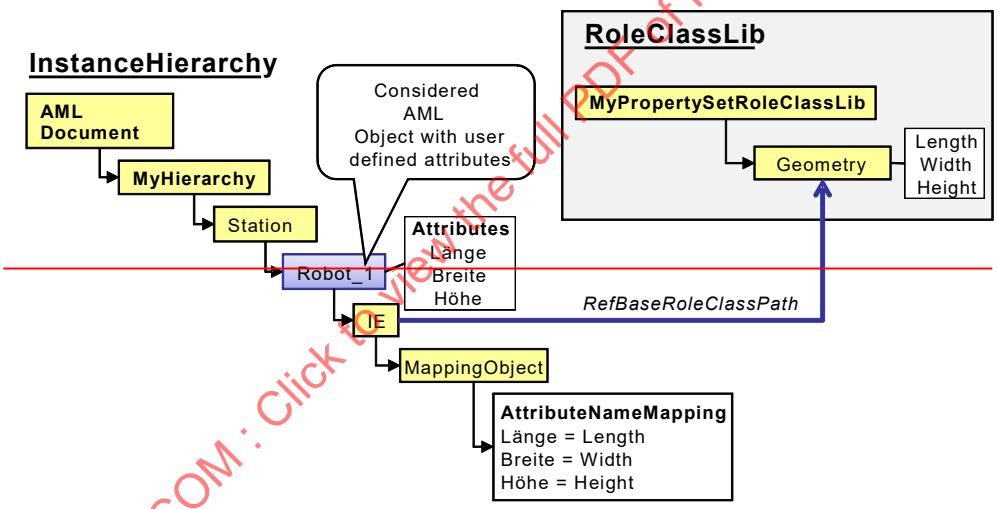
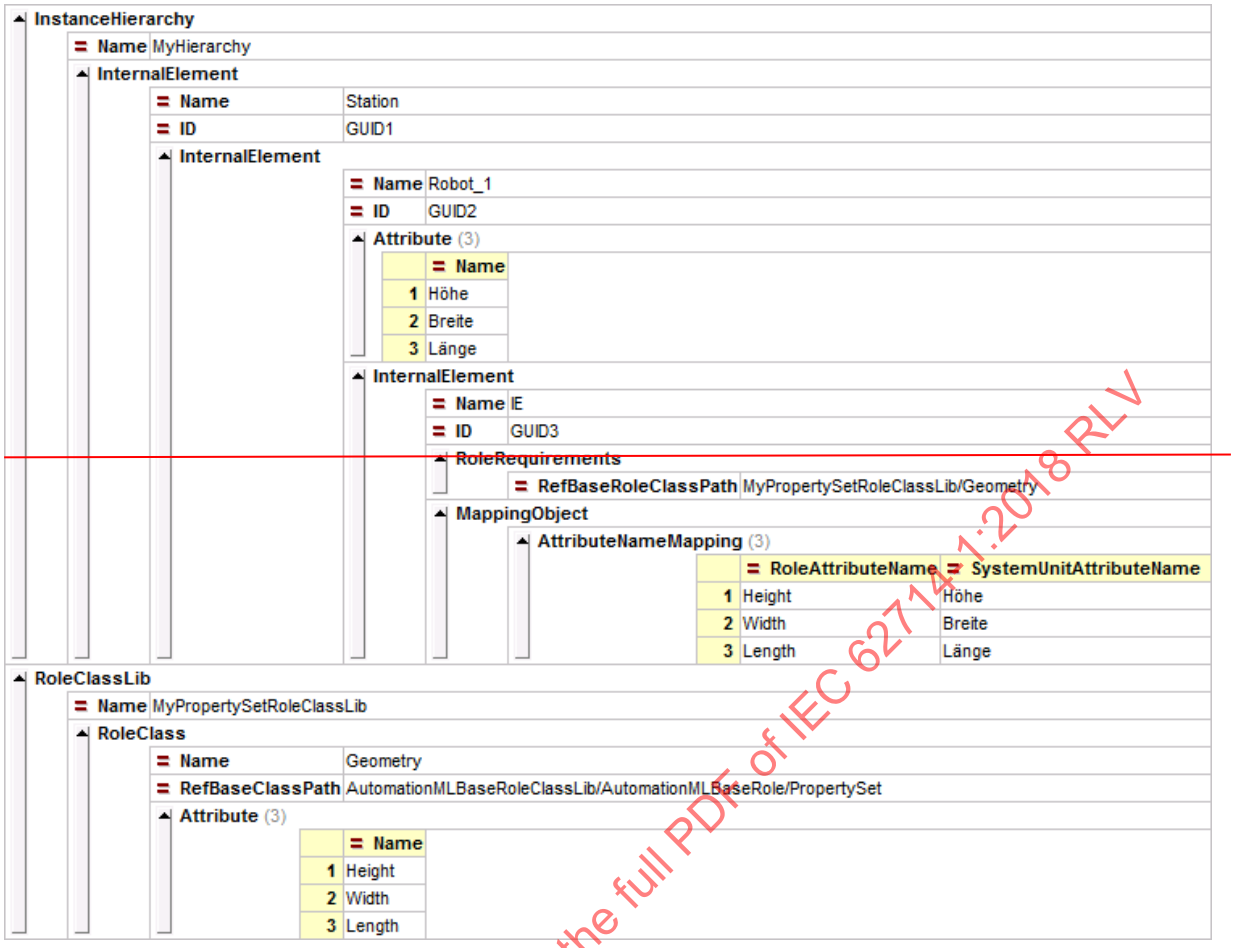


Figure 23 – Example illustrating the PropertySet concept




```
<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.~~

8.5 Splitting of AML top-level data into different documents

According to IEC 62424:~~2008~~ 2016, 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.

8.6 Internationalization, AML multilingual expression

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

The AML multilingual expression concept allows to store textual expressions in different languages as one attribute structure. An informative overview about the multilingual expression concept including examples is provided in A.2.6.

Regarding AML multilingual expressions, the following provisions apply:

- A multilingual text attribute shall be modelled as CAEX Attribute. The value of this attribute shall represent the default expression. The default expression may be used if no specific language is requested or the requested language is not modelled.
- Each language expression of the attribute shall be modelled as nested attribute. Each of these child attributes shall reference the AttributeType "LocalizedAttribute".

- The name of each child-attribute shall be the language of the expression in compliance with RFC 5646, e.g. “en-US”, “de-DE” or “fr-FR” etc. The value of the language attributes shall be the text in the related language.

Additional normative provisions regarding multilingual expressions are provided in 6.5.2.

8.7 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~~ 2016, 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.

8.8 Modelling of structured attribute lists or arrays

In many applications, the storage of lists is needed, e.g. a list of supported frequencies. AML allows the modelling and storage of list attributes and arrays. For the modelling of lists, the following provisions apply:

- A list is a sequence of homogenous items, i.e. all items shall be of the same data type.
- A list is modelled as CAEX Attribute that acts as root node for the list.
- If the list is not ordered, this list attribute shall reference to the AttributeType “ListType”.
- If the list is ordered, this list attribute shall reference to the AttributeType “OrderedListType”.
- The AttributeDataType, Value, DefaultValue and Unit of the list attribute shall be empty.
- The list items shall be modelled as child CAEX attributes of the list attribute.
- All child attributes shall be of the same data type.
- In case of an ordered list, the name of the child attributes shall be represented by an integer number “1”, “2” etc. For better readability, leading zeros can be appended, e.g. “0001”. The integer numbers shall represent the order index of each list item.

NOTE The term integer does not imply a data type.

- In case of a non-ordered list, the names of the child attributes shall be unique across the siblings.
- The child attributes may be again list attributes. This allows modelling arrays. In this case, all child attributes shall be referring to the AttributeType “ListType” or “OrderedListType”.

Subclause A.2.7 provides more information about lists, arrays and explains the modelling by means of examples.

8.9 AML Container

In order to transport an AML project containing multiple AML and other documents, this document supports the Open Packaging Conventions (OPC) as container format. Following the OPC definitions, the following provision apply:

- AML container shall be stored as OPC archive which provides data compression according to ISO/IEC 29500-2.
- AML container shall be either self-contained or environment-connected. Self-contained AML container shall include all related documents which are only locally interlinked with each other. Environment-connected AML container may contain links to URIs to public documents outside of the AML container.

- AML container documents shall have the file extension ".amlx”.
- Following to the OPC convention, this document defines the following relationship types.

Root AML File:

A root AML file is an AML file acting as entry point into the AML container.

Relationshiptype: http://schemas.automationml.org/container/relationship/RootDocument

Mime type: "model/vnd.automationml+xml"

Library AML File:

A library AML file is an AML library which contains one or multiple elements of type Role-ClassLibrary, InterfaceClassLibrary, SystemUnitClassLibrary and/or AttributeTypeLibrary. Similar to the root AML file, the library AML file is an entry point into the AML container. AML containers may contain library files only.

Relationshiptype: http://schemas.automationml.org/container/relationship/Library

Mime type: "model/vnd.automationml+xml“

COLLADA File:

Relationshiptype: http://schemas.automationml.org/container/relationship/Collada

Mime type: "model/vnd.collada+xml"

PLCopenXML File:

Relationshiptype: http://schemas.automationml.org/container/relationship/PLCOpenXML

Mime type: "model/vnd.plcopen+xml"

Any Content:

Relationshiptype: http://schemas.automationml.org/container/relationship/AnyContent

Mime type: "application/x-any" or user-defined according to RFC 2046.

CAEX Schema:

Relationshiptype: http://schemas.automationml.org/container/relationship/CAEXSchema

Mime type: “text/xml”

PLCopenXML Schema:

Relationshiptype:
http://schemas.automationml.org/container/relationship/PCLOpenXMLSchema

Mime type: “text/xml”

COLLADA Schema:

Relationshiptype: http://schemas.automationml.org/container/relationship/ColladaSchema

Mime type: “text/xml”

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 interconnect 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 modelling 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~~. CAEX utilizes AML to interconnect 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.

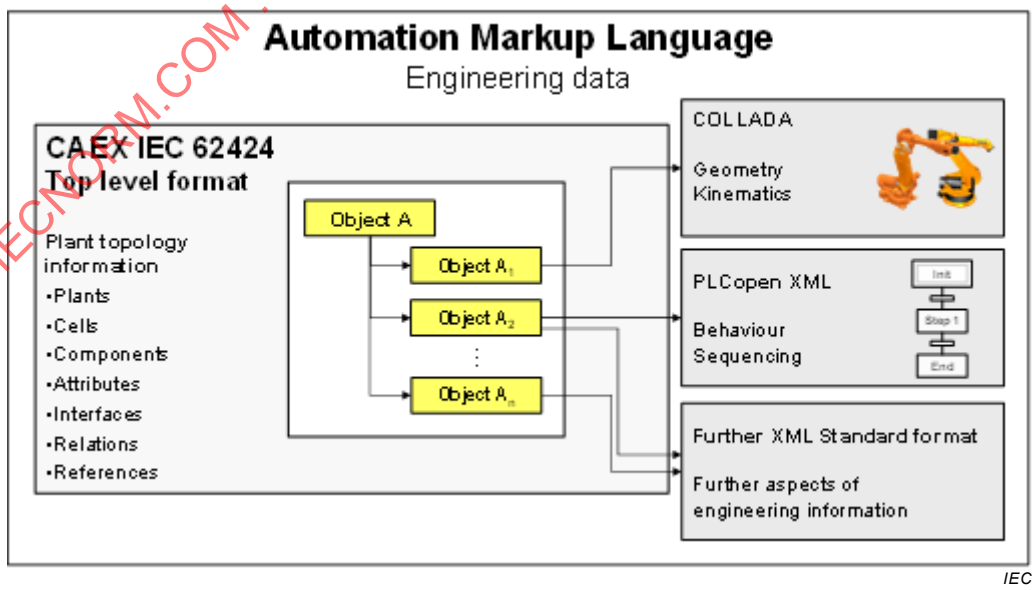


Figure A.1 – AML general architecture

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~~ 2016, 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.1.7 provides an informative overview about the modelling of references and relations with AML. Subclauses 5.5 and 5.6 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 3.1.19).

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~~ IEC 62424:2016, A.2.8. Figure A.2 shows an example of a plant topology of a manufacturing line containing several objects of different hierarchical levels.

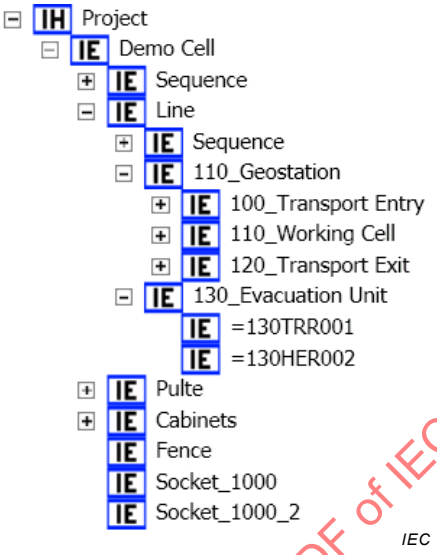


Figure A.2 – Plant topology with AML

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~~ data types for interfaces, roles, system units, attributes 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~~ elements and thus ~~serve~~ enable the automatic semantic interpretation of ~~AML objects~~ userdefined SystemUnitClasses or InternalElements. Subclause 6.4 specifies the basic AML-RoleClassLib with a set of abstract RoleClasses which are dedicated to general automation systems. Subclause 7.5 specifies the modelling of user-defined role classes. ~~It is intended to specify~~ Further role libraries are defined in IEC 62714-2.
- **SystemUnits and SystemUnitClassLib**: The SystemUnitClassLib is used to ~~store~~ model vendor specific ~~AML~~ components and their internal structure as classes. Subclause 7.6 specifies architecture rules for the definition of SystemUnitClasses. AML does not predefine a certain SystemUnitClassLib or SystemUnitClass.
- **AttributeTypes and AttributeTypeLib**: The AttributeTypeLib is used to model vendor specific Attribute Types or standardized Attribute Types. Attributes which are derived from an AttributeType inherit their semantics, even when they have a different name. This is the basis for language independence. Attribute type libraries can be utilized as basis for semantic libraries and serve for the automatic interpretation of vendor specific attributes.
- **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 7.7 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~~ IEC 62424:2016 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.

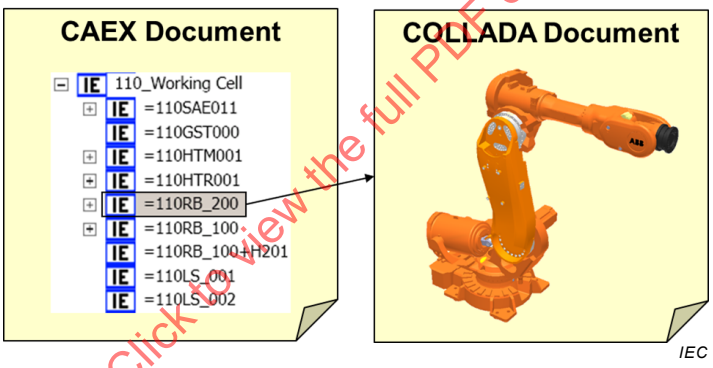


Figure A.3 – Reference from CAEX to a COLLADA document

The reference is modelled by means of a CAEX interface derived from the standard AML interface class "COLLADAInterface" specified in 5.6 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.

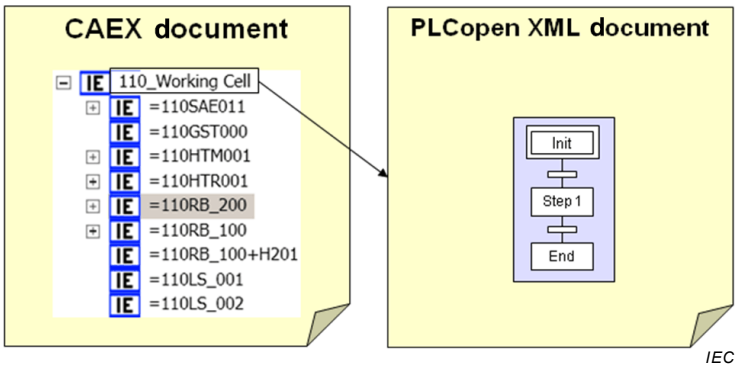


Figure A.4 – Reference from a CAEX to a PLCopen XML document

A.1.5 Referencing documents outside of the scope of IEC 62714

This document defines a mechanism which allows to reference external documents which are not in the scope of IEC 62714 (e.g. documents, pdf files, Excel sheets, XML files outside of the scope of IEC 62714, etc.). Figure A.5 and Figure A.6 illustrate how to model this.

In this example, the object “TemperatureSensor” should reference an external file “example.xml”. To model this reference, the InternalElement “TemperatureSensor” has a nested InternalElement “ParameterDocument” of type “ExternalData” representing the external document. Its attribute “DocLang” models the language of the document. The InternalElement “ParameterDocument” contains a CAEX ExternalInterface of type ExternalDataReference. Its attributes refURI and MIMEType define the URI of the document and the document type, in this case it is of the MIMEType “application/xml”. Normative provisions are defined in 5.6.5.

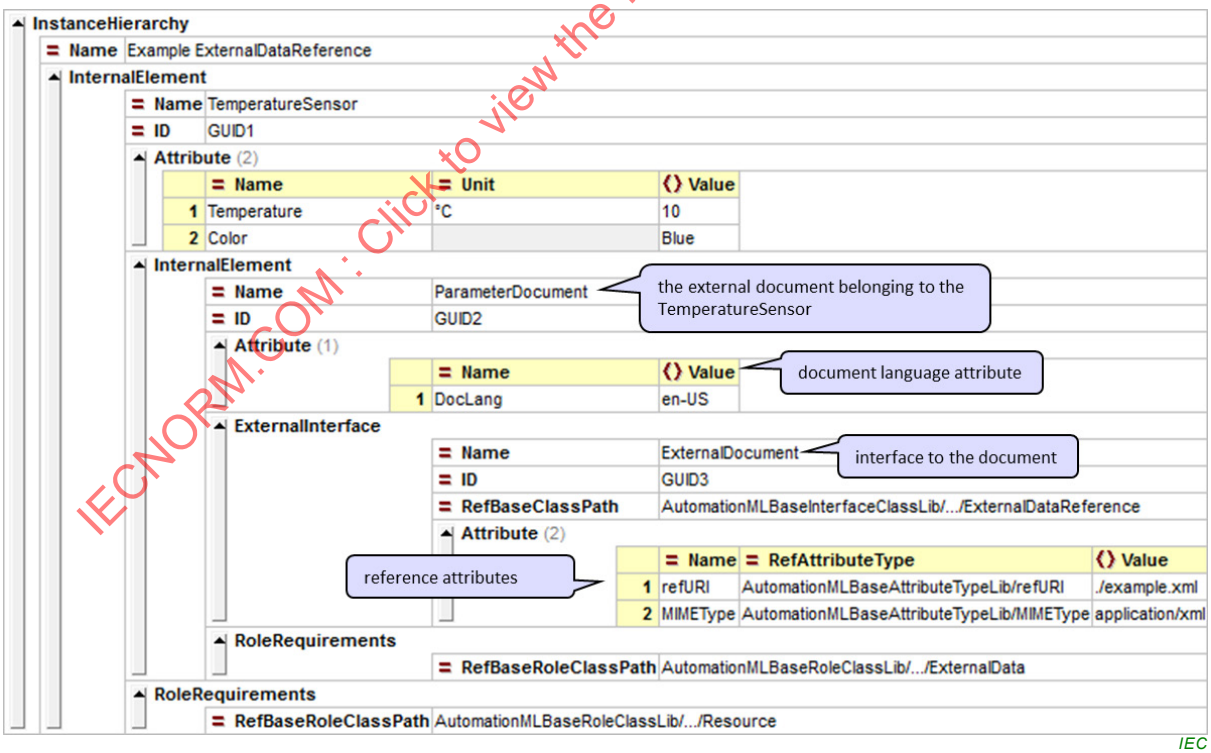


Figure A.5 – Example of referencing an external document

```
<InstanceHierarchy Name="Example ExternalDataReference">
  <InternalElement Name="TemperatureSensor" ID="GUID1">
    <Attribute Name="Temperature" Unit="°C">
      <Value>10</Value>
    </Attribute>
    <Attribute Name="Color">
      <Value>Blue</Value>
    </Attribute>
    <InternalElement Name="ParameterDocument" ID="GUID2">
      <Attribute Name="DocLang">
        <Value>en-US</Value>
      </Attribute>
      <ExternalInterface Name="ExternalDocument" ID="GUID3" RefBaseClassPath="AutomationMLBaseInterfaceClassLib/.../ExternalDataReference">
        <Attribute Name="refURI" RefAttributeType="AutomationMLBaseAttributeTypeLib/refURI">
          <Value>./example.xml</Value>
        </Attribute>
        <Attribute Name="MIMEType" RefAttributeType="AutomationMLBaseAttributeTypeLib/MIMEType">
          <Value>application/xml</Value>
        </Attribute>
      </ExternalInterface>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../ExternalData"/>
    </InternalElement>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
  </InternalElement>
</InstanceHierarchy>
```

IEC

Figure A.6 – XML text of the example for referencing an external document

A.1.6 Interlinking CAEX attributes and attributes in external documents

Figure A.7 and Figure A.8 illustrate how to model the interlinking between a CAEX attribute and an item in an external document. This example is an extension of the example shown in A.1.5: the Attribute “Temperature” should be referenced to an item in the external document “example.xml”.

In addition to the example shown in A.1.5/Figure A.5., the ExternalInterface “ExternalDocument” models a further attribute of type “AssociatedExternalValue”. This attribute contains 3 nested attributes.

- The attribute “refCAEXAttribute” mirrors the desired CAEX attribute. This attribute has no value.
- The attribute “refURI” references the external attribute. It references the same external document as the refURI attribute of the ExternalInterface: the “example.xml”.
- The attribute “Direction” provides the direction of the referencing, in this example the external document value is leading, CAEX is the consumer of the external value.

Normative provisions are defined in 5.6.6.

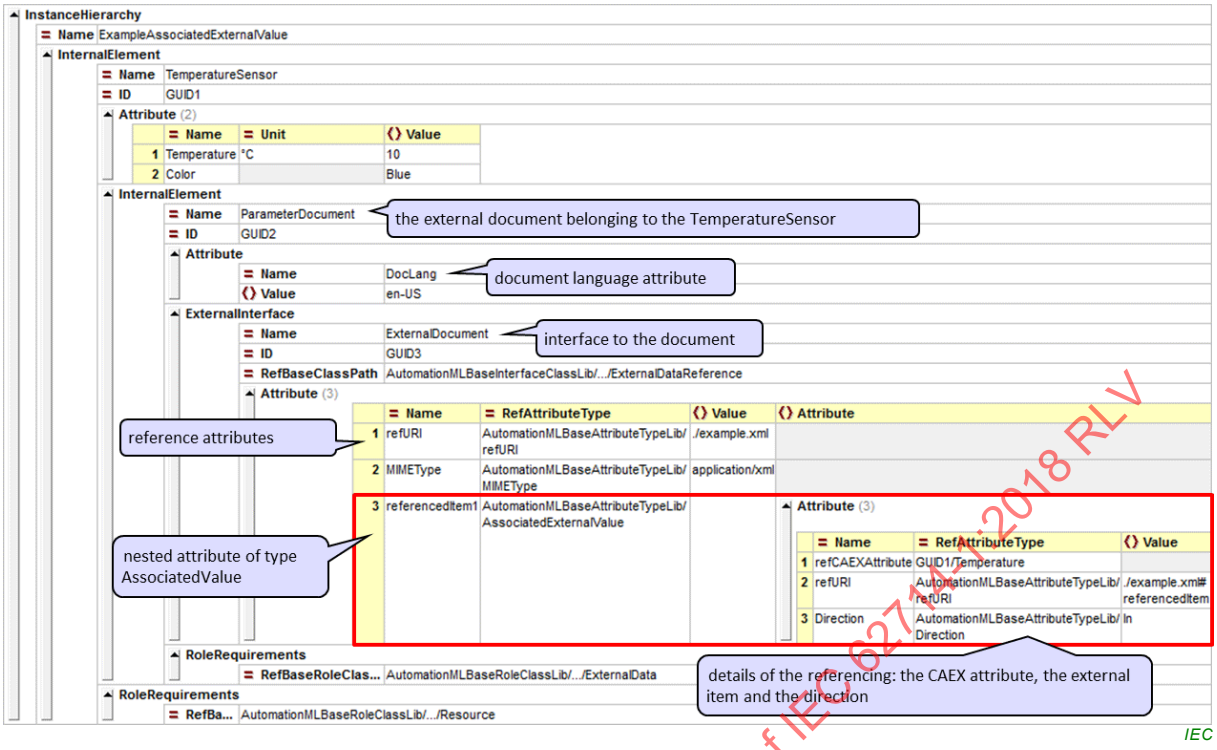


Figure A.7 – Example of referencing a CAEX attribute to an item in an external document

```
<InstanceHierarchy Name="ExampleAssociatedExternalValue">
  <InternalElement Name="TemperatureSensor" ID="GUID1">
    <Attribute Name="Temperature" Unit="°C">
      <Value>10</Value>
    </Attribute>
    <Attribute Name="Color">
      <Value>Blue</Value>
    </Attribute>
    <InternalElement Name="ParameterDocument" ID="GUID2">
      <Attribute Name="DocLang">
        <Value>en-US</Value>
      </Attribute>
      <ExternalInterface Name="ExternalDocument" ID="GUID3" RefBaseClassPath="AutomationMLBaseInterfaceClassLib/.../ExternalDataReference">
        <Attribute Name="refURI" RefAttributeType="AutomationMLBaseAttributeTypeLib/refURI">
          <Value>./example.xml</Value>
        </Attribute>
        <Attribute Name="MIMEType" RefAttributeType="AutomationMLBaseAttributeTypeLib/MIMEType">
          <Value>application/xml</Value>
        </Attribute>
        <Attribute Name="referencedItem1" RefAttributeType="AutomationMLBaseAttributeTypeLib/AssociatedExternalValue">
          <Value/>
          <Attribute Name="refCAEXAttribute" RefAttributeType="GUID1/Temperature"/>
          <Attribute Name="refURI" RefAttributeType="AutomationMLBaseAttributeTypeLib/refURI">
            <Value>./example.xml#referencedItem</Value>
          </Attribute>
          <Attribute Name="Direction" RefAttributeType="AutomationMLBaseAttributeTypeLib/Direction">
            <Value>In</Value>
          </Attribute>
        </Attribute>
      </ExternalInterface>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../ExternalData"/>
    </InternalElement>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
  </InternalElement>
</InstanceHierarchy>
```

IEC

Figure A.8 – XML text of the example for referencing a CAEX attribute to an item in an external document

A.1.7 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~~ CAEX supports the following relations according to IEC 62424:2016, A.2.8.

- 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
  - inheritance relations between AttributeTypes4
- class-instance-relations (see 5.5.2)
  - 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
  - relations between an AttributeType and an attribute
- instance-instance-relations (see 5.5.3)
  - relations between AML objects
  - relations between published externally stored data

Figure A.9 presents the mentioned relation types supported by AML by means of an example.

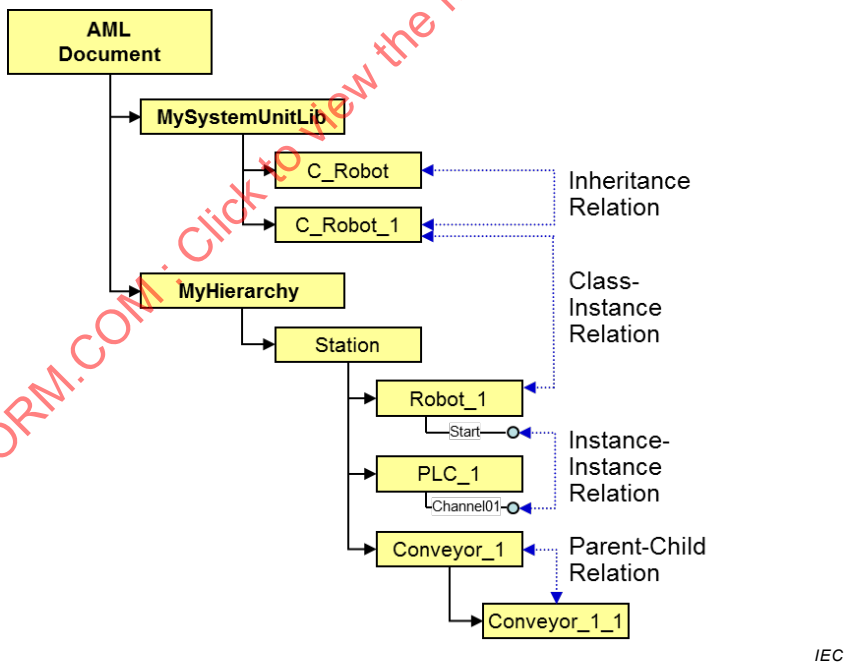


Figure A.9 – Relations in AML

Figure A.10 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.11 shows the corresponding XML text of the AML library.

Figure A.12 shows the XML text of the InstanceHierarchy.

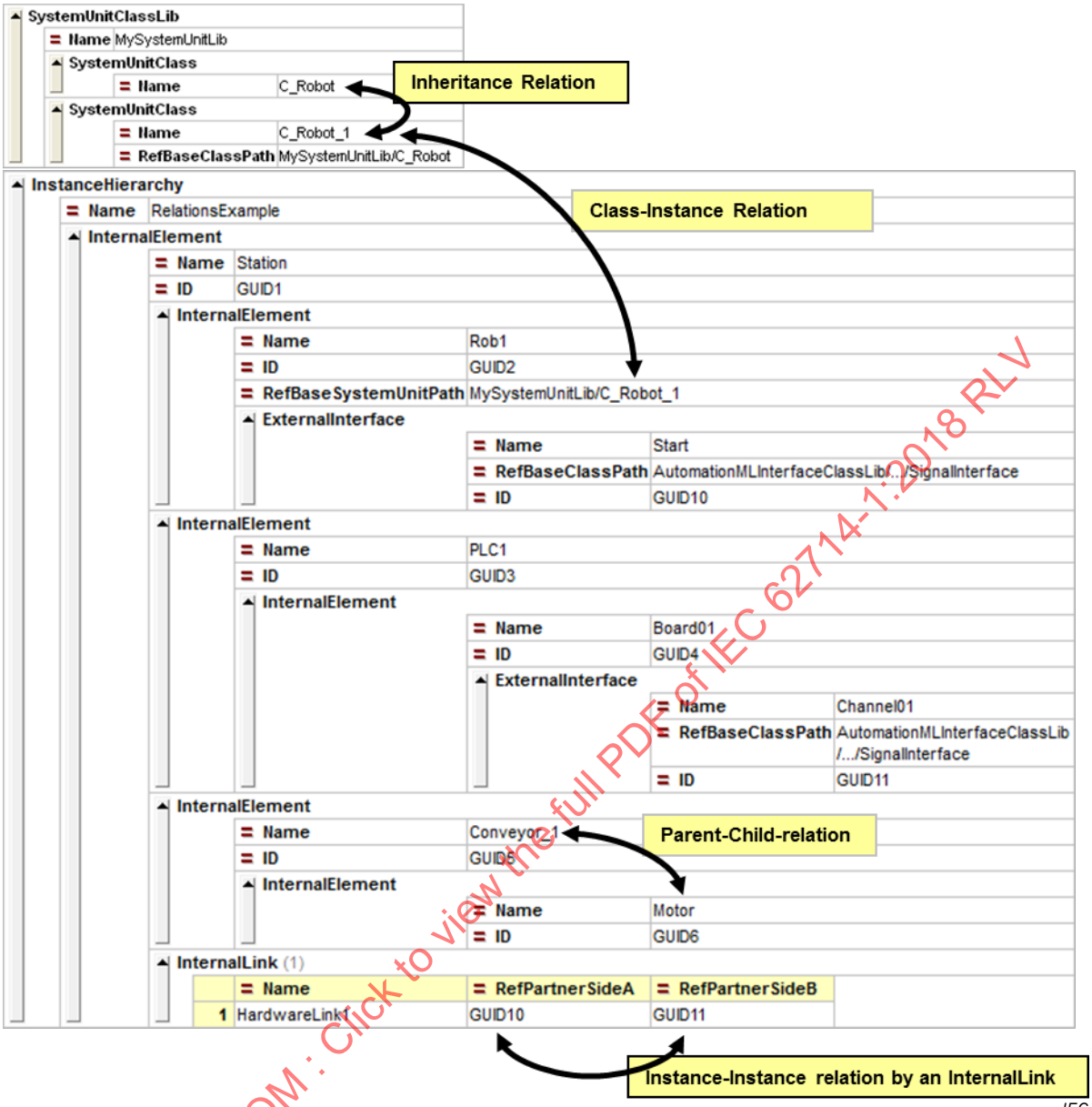


Figure A.10 – XML description of the relations example

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

IEC

Figure A.11 – 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" ID="GUID10"/>
    </InternalElement>
    <InternalElement Name="PLC1" ID="GUID3">
      <InternalElement Name="Board01" ID="GUID4">
        <ExternalInterface Name="Channel01" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface" ID="GUID11"/>
      </InternalElement>
    </InternalElement>
    <InternalElement Name="Conveyor_1" ID="GUID5">
      <InternalElement Name="Motor" ID="GUID6"/>
    </InternalElement>
    <InternalLink Name="HardwareLink1" RefPartnerSideA="GUID10" RefPartnerSideB="GUID11"/>
  </InternalElement>
</InstanceHierarchy>
```

IEC

Figure A.12 – 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.
AML multilingual expression	AML multilingual expressions allow to store different languages for a textual expression.
List attribute and attribute arrays	AML allows modelling of attribute lists or arrays of attributes.

A.2.2 AML Port concept

A.2.2.1 Concept description

An AML ~~object~~ interface that groups a number of nested interfaces (see Figure A.13). 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~~ InterfaceClass “Port" (see 6.3.4). Normative provisions are specified in 8.2.

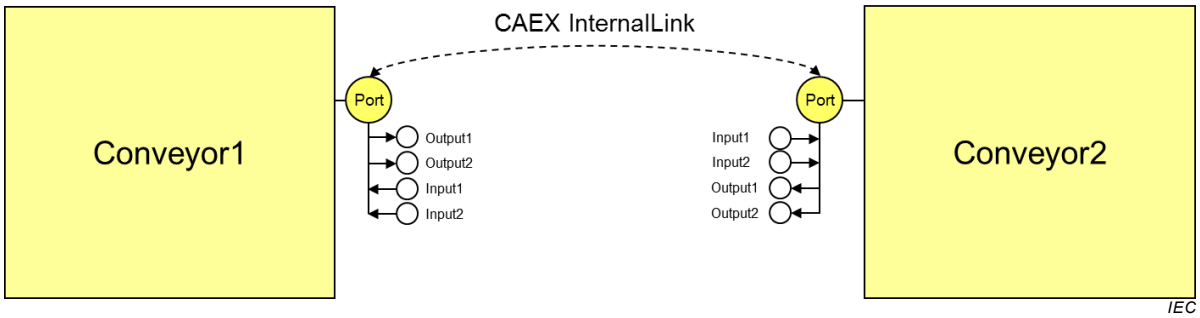


Figure A.13 – Port concept

A.2.2.2 Example

Figure A.14 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 “ConnectionPoint” derived from the AML InterfaceClass “PortConnector”.~~ Both sub-objects have a complex interface containing sub-interfaces. This is modelled by means of each a Port interface, derived from the AML standard InterfaceClass “Port”, and comprising a collection of nested interfaces. The 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~~ Ports are connected. In addition to this concept, AML allows modelling and storage of each individual link between the sub-interfaces.

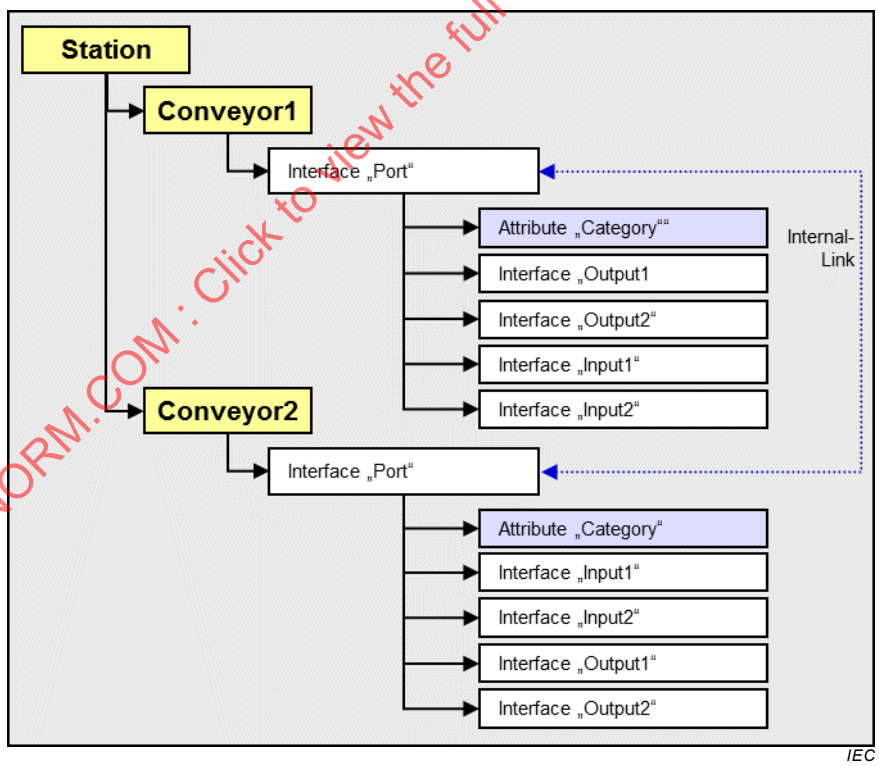


Figure A.14 – Example describing the AML Port concept

Figure A.15 and Figure A.16 describe the AML implementation of the example system described in Figure A.14.

- InstanceHierarchy
  - Name: PortExample AML 2.10
  - InternalElement
    - Name: Station
    - ID: GUID1
    - InternalElement
      - Name: Conveyor1
      - ID: GUID2
      - ExternalInterface
        - Name: Port
        - ID: GUID3
        - RefBaseClassPath: AutomationMLInterfaceClassLib/.../Port
        - Attribute
          - Name: Direction
          - AttributeDataType: xs:string
          - RefAttributeType: AutomationMLBaseAttributeTypeLib/Direction
          - Value: Out
        - Attribute
          - Name: Category
          - AttributeDataType: xs:string
          - RefAttributeType: AutomationMLBaseAttributeTypeLib/Category
          - Value: MaterialFlow
        - ExternalInterface (4)

	Name	RefBaseClassPath	ID
1	Output1	MyInterfaceClassLib/SignalInterface	GUID5
2	Output2	MyInterfaceClassLib/SignalInterface	GUID6
3	Input1	MyInterfaceClassLib/SignalInterface	GUID7
4	Input2	MyInterfaceClassLib/SignalInterface	GUID8

      - RoleRequirements
        - RefBaseRoleClassPath: AutomationMLRoleClassLib/AutomationMLBaseRole/Resource
    - InternalElement
      - Name: Conveyor2
      - ID: GUID4
      - ExternalInterface
        - Name: Port
        - ID: GUID10
        - RefBaseClassPath: AutomationMLInterfaceClassLib/.../Port
        - Attribute
          - Name: Direction
          - AttributeDataType: xs:string
          - RefAttributeType: AutomationMLBaseAttributeTypeLib/Direction
          - Value: In
        - Attribute
          - Name: Category
          - AttributeDataType: xs:string
          - RefAttributeType: AutomationMLBaseAttributeTypeLib/Category
          - Value: MaterialFlow
        - ExternalInterface (4)

	Name	RefBaseClassPath	ID
1	Input1	MyInterfaceClassLib/SignalInterface	GUID12
2	Input2	MyInterfaceClassLib/SignalInterface	GUID13
3	Output1	MyInterfaceClassLib/SignalInterface	GUID14
4	Output2	MyInterfaceClassLib/SignalInterface	GUID15

      - RoleRequirements
        - RefBaseRoleClassPath: AutomationMLRoleClassLib/AutomationMLBaseRole/Resource
    - InternalLink (1)

	Name	RefPartnerSideA	RefPartnerSideB
1	L1	GUID3	GUID10

IEC

Figure A.15 – XML description of the AML Port concept

```
<InstanceHierarchy Name="PortExample AML 2.10">
  <InternalElement Name="Station" ID="GUID1">
    <InternalElement Name="Conveyor1" ID="GUID2">
      <ExternalInterface Name="Port" ID="GUID3" RefBaseClassPath="AutomationMLInterfaceClassLib/.../Port">
        <Attribute Name="Direction" AttributeDataType="xs:string" RefAttributeType="AutomationMLBaseAttributeTypeLib/Direction">
          <Value>Out</Value>
        </Attribute>
        <Attribute Name="Category" AttributeDataType="xs:string" RefAttributeType="AutomationMLBaseAttributeTypeLib/Category">
          <Value>MaterialFlow</Value>
        </Attribute>
        <ExternalInterface Name="Output1" RefBaseClassPath="MyInterfaceClassLib/SignaleInterface" ID="GUID5"/>
        <ExternalInterface Name="Output2" RefBaseClassPath="MyInterfaceClassLib/SignaleInterface" ID="GUID6"/>
        <ExternalInterface Name="Input1" RefBaseClassPath="MyInterfaceClassLib/SignaleInterface" ID="GUID7"/>
        <ExternalInterface Name="Input2" RefBaseClassPath="MyInterfaceClassLib/SignaleInterface" ID="GUID8"/>
      </ExternalInterface>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource"/>
    </InternalElement>
    <InternalElement Name="Conveyor2" ID="GUID4">
      <ExternalInterface Name="Port" ID="GUID10" RefBaseClassPath="AutomationMLInterfaceClassLib/.../Port">
        <Attribute Name="Direction" AttributeDataType="xs:string" RefAttributeType="AutomationMLBaseAttributeTypeLib/Direction">
          <Value>In</Value>
        </Attribute>
        <Attribute Name="Category" AttributeDataType="xs:string" RefAttributeType="AutomationMLBaseAttributeTypeLib/Category">
          <Value>MaterialFlow</Value>
        </Attribute>
        <ExternalInterface Name="Input1" RefBaseClassPath="MyInterfaceClassLib/SignaleInterface" ID="GUID12"/>
        <ExternalInterface Name="Input2" RefBaseClassPath="MyInterfaceClassLib/SignaleInterface" ID="GUID13"/>
        <ExternalInterface Name="Output1" RefBaseClassPath="MyInterfaceClassLib/SignaleInterface" ID="GUID14"/>
        <ExternalInterface Name="Output2" RefBaseClassPath="MyInterfaceClassLib/SignaleInterface" ID="GUID15"/>
      </ExternalInterface>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource"/>
    </InternalElement>
    <InternalLink Name="L1" RefPartnerSideA="GUID3" RefPartnerSideB="GUID10"/>
  </InternalElement>
</InstanceHierarchy>
```

IEC

Figure A.16 – 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 SystemUnitClass "myPortClass".~~

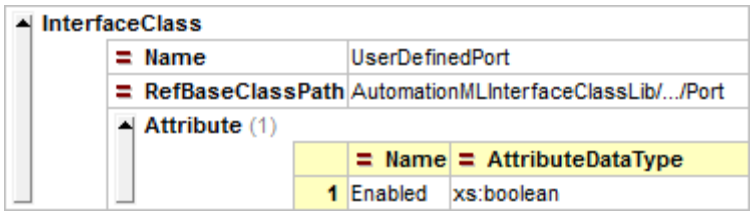
SystemUnitClass			
= Name	myPortClass		
Attribute (2)			
		= Name	{} Value
	1	Direction	InOut
	2	Category	MaterialFlow
ExternalInterface			
		= Name	ConnectionPoint
		= RefBaseClassPath	AutomationMLInterfaceClassLib/.../PortConnector
SupportedRoleClass			
		= RefRoleClassPath	AutomationMLBaseRoleClassLib/.../Port

```
<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.2.2.3 Modelling a Port as user-defined Port

Ports can be extended by deriving an InterfaceClass. The following example in Figure A.17 depicts an XML description of a derived InterfaceClass “myPortInterface”. This class inherits all attributes of the standard class “Port” and adds a new attribute “Enabled”.




```
<InterfaceClass Name="UserDefinedPort" RefBaseClassPath="AutomationMLInterfaceClassLib/.../Port">
  <Attribute Name="Enabled" AttributeDataType="xs:boolean"/>
</InterfaceClass>
```

IEC

Figure A.17 – Definition of a user-defined AML Port class “UserDefinedPort”

A.2.3 AML Facet concept

A.2.3.1 Concept description

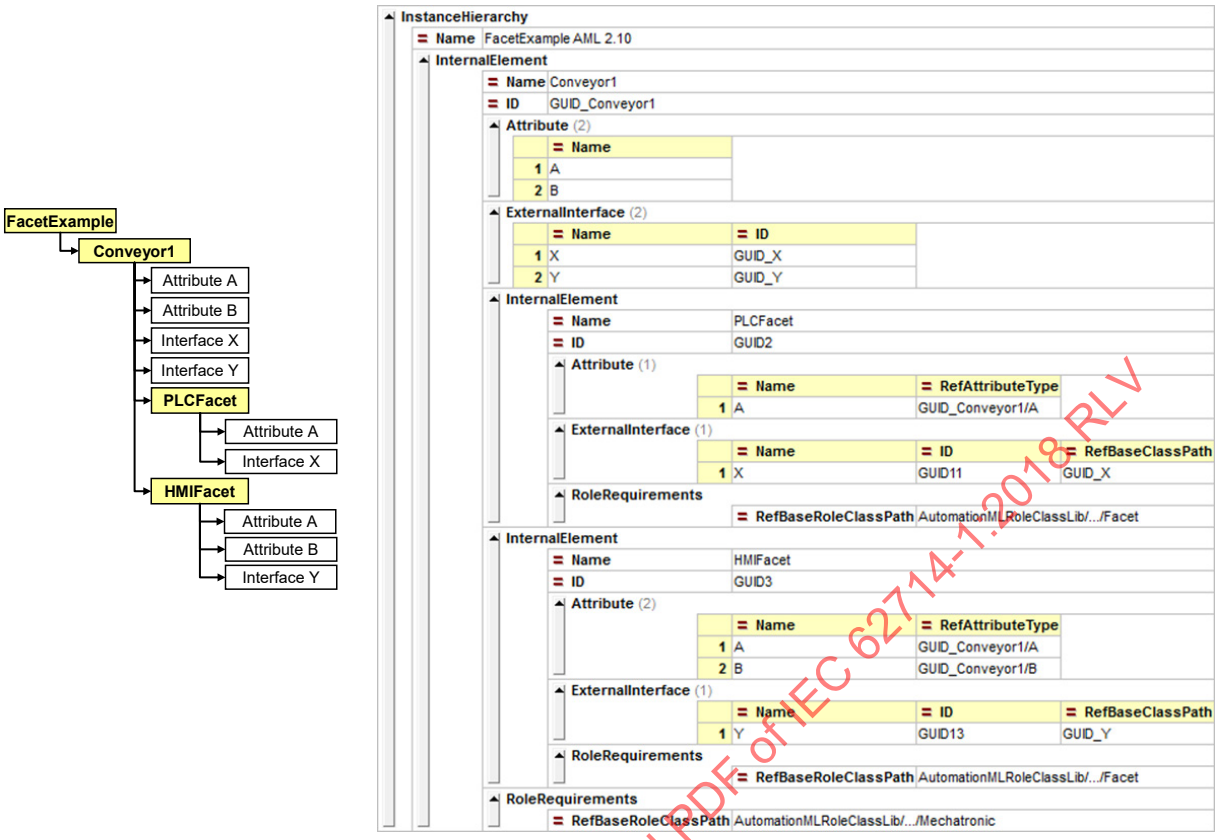
A Facet is an AML object providing a sub-view on attributes or interfaces of the parent AML object. This concept serves for the storage of different configuration settings such as HMI or PLC related data and allows the automation of several control engineering steps. For this, AML defines the AML RoleClass “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.18 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.



IEC

Figure A.18 – AML Facet example

```
<InstanceHierarchy Name="FacetExample AML 2.10">
  <InternalElement Name="Conveyor1" ID="GUID_Conveyor1">
    <Attribute Name="A"/>
    <Attribute Name="B"/>
    <ExternalInterface Name="X" ID="GUID_X"/>
    <ExternalInterface Name="Y" ID="GUID_Y"/>
    <InternalElement Name="PLCFacet" ID="GUID2">
      <Attribute Name="A" RefAttributeType="GUID_Conveyor1/A"/>
      <ExternalInterface Name="X" ID="GUID11" RefBaseClassPath="GUID_X"/>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../Facet"/>
    </InternalElement>
    <InternalElement Name="HMIFacet" ID="GUID3">
      <Attribute Name="A" RefAttributeType="GUID_Conveyor1/A"/>
      <Attribute Name="B" RefAttributeType="GUID_Conveyor1/B"/>
      <ExternalInterface Name="Y" ID="GUID13" RefBaseClassPath="GUID_Y"/>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../Facet"/>
    </InternalElement>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../Mechatronic"/>
  </InternalElement>
</InstanceHierarchy>
```

IEC

Figure A.19 – XML text of the AML Facet example

A.2.4 AML Group concept

A.2.4.1 Concept description

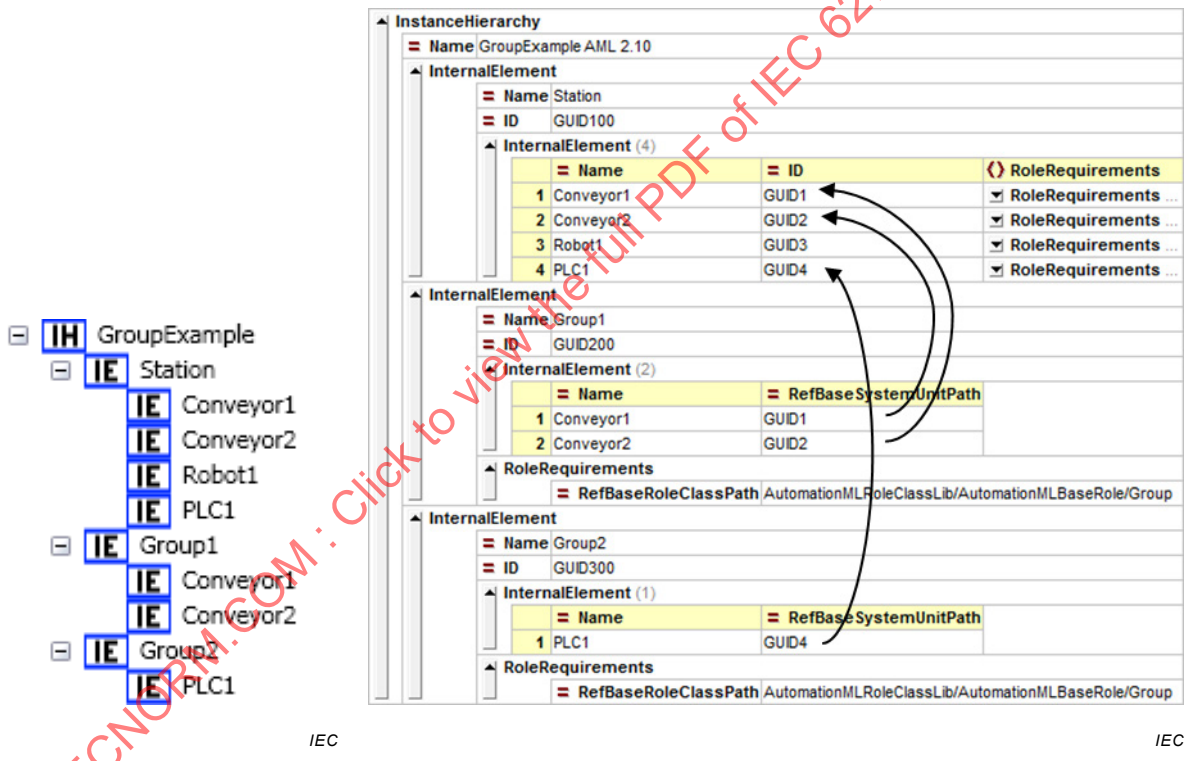
The AML Group concept allows separating structure information from instance information. Since different engineering tools in a heterogeneous tool landscape may require different

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.20a) 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~~ IEC 62424:2016, A.2.8.7, CAEX provides the storage of such crossed structures. Figure A.20b) gives an AML implementation of this example and Figure A.21 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.20 – AML Group example

```
<InstanceHierarchy Name="GroupExample AML 2.10">
  <InternalElement Name="Station" ID="GUID100">
    <InternalElement Name="Conveyor1" ID="GUID1">
      <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../Transport/Rollerbed"/>
    </InternalElement>
    <InternalElement Name="Conveyor2" ID="GUID2">
      <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../Rollerbed"/>
    </InternalElement>
    <InternalElement Name="Robot1" ID="GUID3">
      <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../Robot"/>
    </InternalElement>
    <InternalElement Name="PLC1" ID="GUID4">
      <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../PLC"/>
    </InternalElement>
  </InternalElement>
  <InternalElement Name="Group1" ID="GUID200">
    <InternalElement Name="Conveyor1" RefBaseSystemUnitPath="GUID1"/>
    <InternalElement Name="Conveyor2" RefBaseSystemUnitPath="GUID2"/>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Group"/>
  </InternalElement>
  <InternalElement Name="Group2" ID="GUID300">
    <InternalElement Name="PLC1" RefBaseSystemUnitPath="GUID4"/>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Group"/>
  </InternalElement>
</InstanceHierarchy>
```

IEC

Figure A.21 – XML text for the AML Group example

A.2.4.3 Combination of the Group and Facet concept

Figure A.22 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.

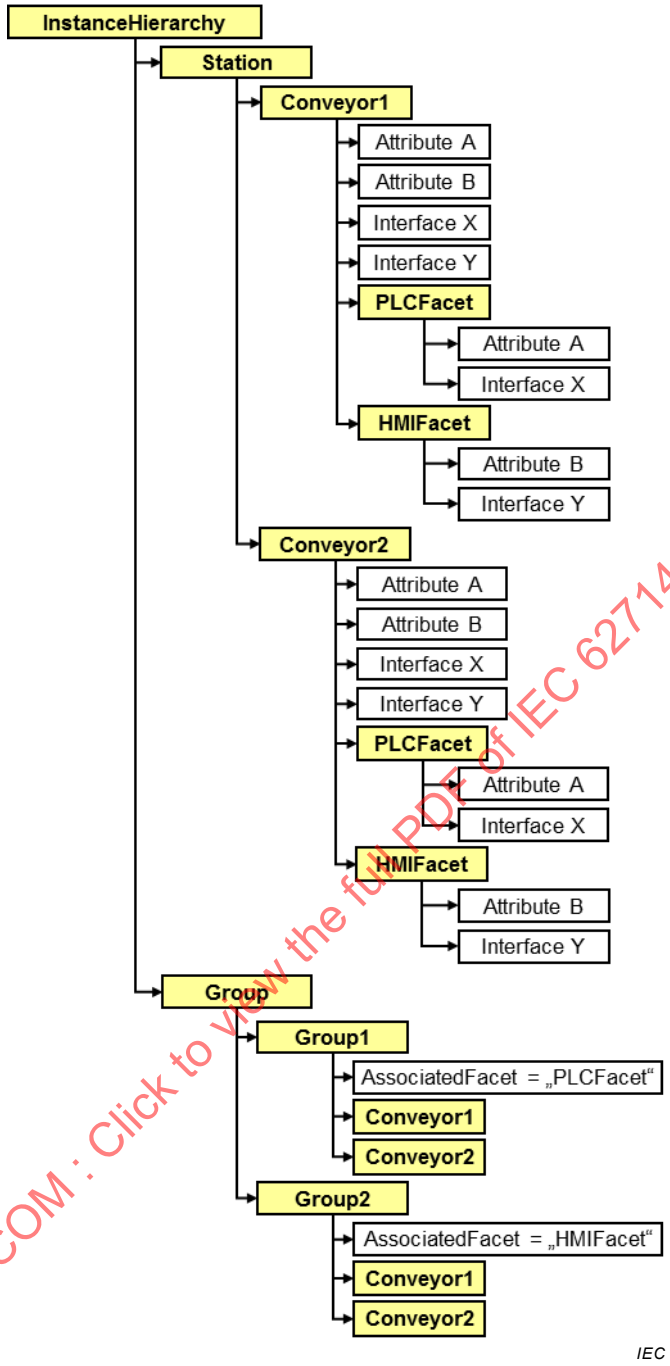


Figure A.22 – Combination of the Facet and Group concept

Figure A.23 shows the corresponding XML text related to the example shown in Figure A.22.

```
<InstanceHierarchy Name="FacetGroupCombination AML 2.10">
  <InternalElement Name="Station" ID="GUID0">
    <InternalElement Name="Conveyor1" ID="GUID1">
      <Attribute Name="A"/>
      <Attribute Name="B"/>
      <ExternalInterface Name="X" ID="GUID_X"/>
      <ExternalInterface Name="Y" ID="GUID_Y"/>
      <InternalElement Name="PLCFacet" ID="GUID2">
        <Attribute Name="A" RefAttributeType="GUID1/A"/>
        <ExternalInterface Name="X" ID="GUID11" RefBaseClassPath="GUID_X"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../Facet"/>
      </InternalElement>
      <InternalElement Name="HMIFacet" ID="GUID3">
        <Attribute Name="B" RefAttributeType="GUID1/B"/>
        <ExternalInterface Name="Y" ID="GUID12" RefBaseClassPath="GUID_Y"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../Facet"/>
      </InternalElement>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../Transport/Rollerbed"/>
    </InternalElement>
    <InternalElement Name="Conveyor2" ID="GUID4">
      <Attribute Name="A"/>
      <Attribute Name="B"/>
      <ExternalInterface Name="X" ID="GUID_X2"/>
      <ExternalInterface Name="Y" ID="GUID_Y2"/>
      <InternalElement Name="PLCFacet" ID="GUID5">
        <Attribute Name="A" RefAttributeType="GUID4/A"/>
        <ExternalInterface Name="X" ID="GUID13" RefBaseClassPath="GUID_X2"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../Facet"/>
      </InternalElement>
      <InternalElement Name="HMIFacet" ID="GUID6">
        <Attribute Name="B" RefAttributeType="GUID4/B"/>
        <ExternalInterface Name="Y" ID="GUID14" RefBaseClassPath="GUID_Y2"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../Facet"/>
      </InternalElement>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../Transport/Rollerbed"/>
    </InternalElement>
    <InternalElement Name="Group" ID="GUID7">
      <InternalElement Name="Group1" ID="GUID8">
        <Attribute Name="AssociatedFacet" RefAttributeType="AutomationMLBaseAttributeTypeLib/AssociatedFacet">
          <Value>HMIFacet</Value>
        </Attribute>
        <InternalElement Name="Conveyor1" RefBaseSystemUnitPath="GUID1"/>
        <InternalElement Name="Conveyor2" RefBaseSystemUnitPath="GUID4"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../Group"/>
      </InternalElement>
      <InternalElement Name="Group2" ID="GUID9">
        <Attribute Name="AssociatedFacet" RefAttributeType="AutomationMLBaseAttributeTypeLib/AssociatedFacet">
          <Value>PLCFacet</Value>
        </Attribute>
        <InternalElement Name="Conveyor1" RefBaseSystemUnitPath="GUID1"/>
        <InternalElement Name="Conveyor2" RefBaseSystemUnitPath="GUID4"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../Group"/>
      </InternalElement>
    </InternalElement>
  </InternalElement>
</InstanceHierarchy>
```

IEC

Figure A.23 – 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.24 illustrates this generic HMI template of a conveyor.

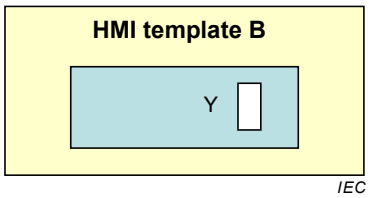


Figure A.24 – Generic HMI template "B" visualizing a process variable "Y" of a conveyor

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~~ extracts the HMI relevant information of ~~each of~~ both conveyors, ~~can identify~~ identifies the corresponding HMI template and ~~can~~ associates the correct signals to ~~be~~ visualize~~d~~.
Figure A.25 represents the resulting HMI.

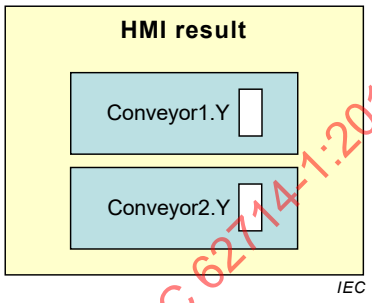


Figure A.25 – Generated HMI result "B" visualizing both conveyors with individual process variables

~~**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.~~

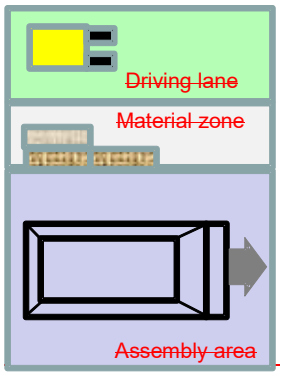


Figure A.22 – PropertySet example

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.

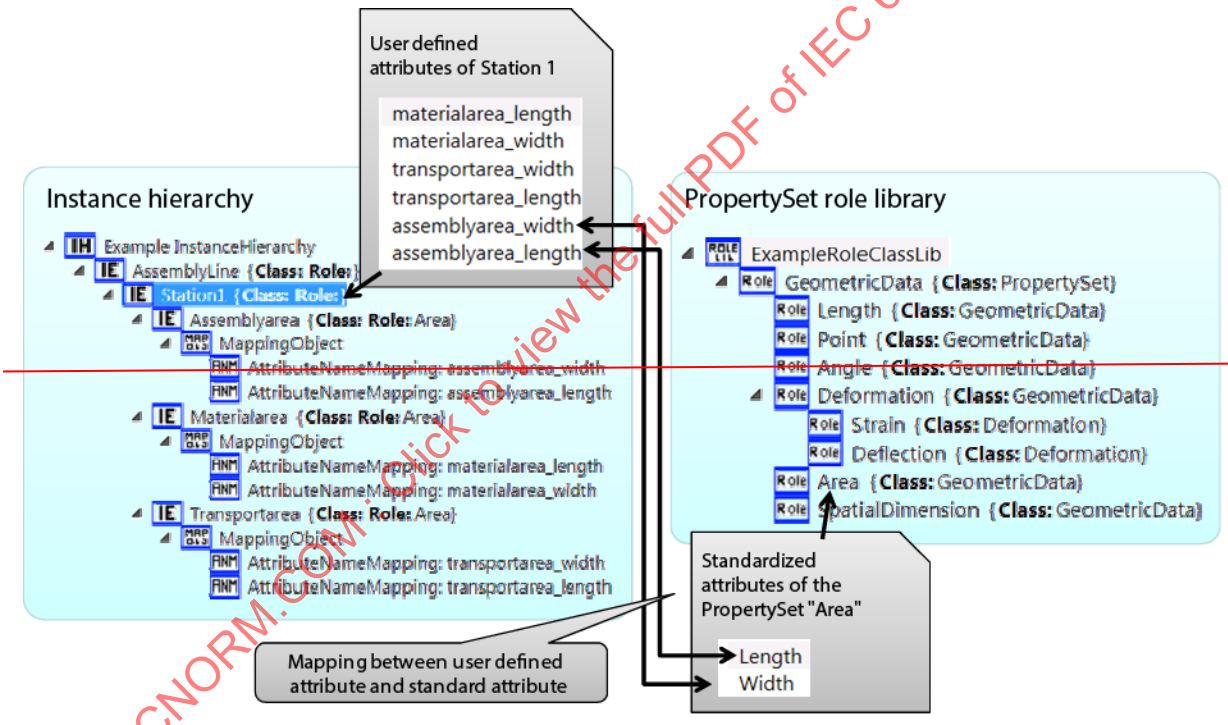


Figure A.23 – PropertySet example

```
<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.2.5 Process-Product-Resource concept

A.2.5.1 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.

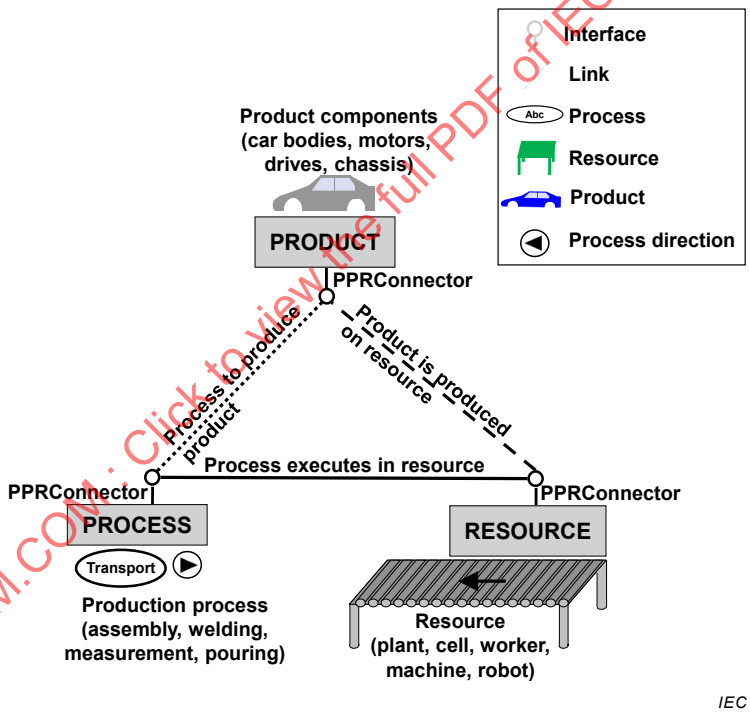


Figure A.26 – Base elements of the Product-Process-Resource concept

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 5.5.3). Thus, resources can be linked to products which they can manipulate.

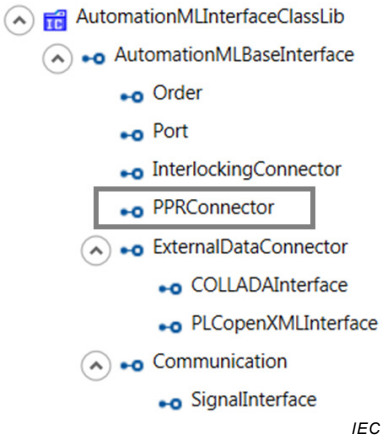


Figure A.27 – PPRConnector interface

A.2.5.2 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.

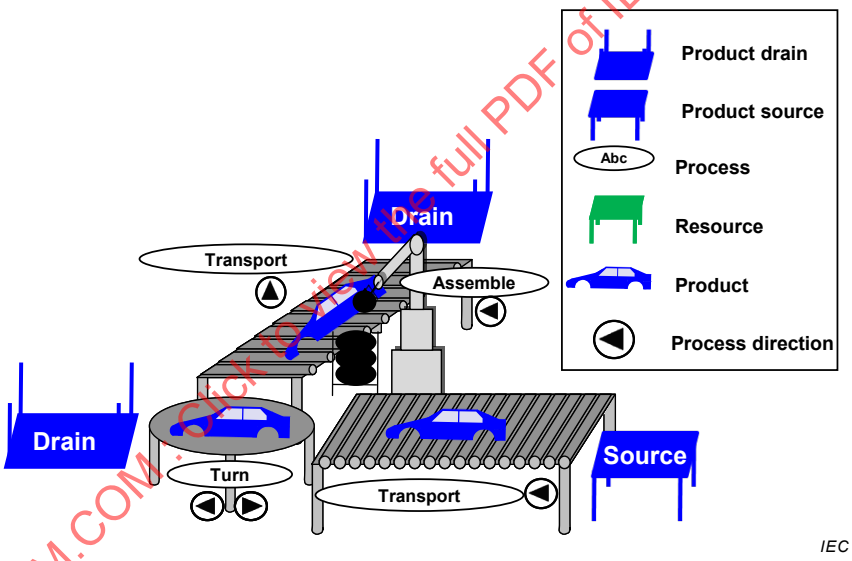


Figure A.28 – Example for the Product-Process-Resource concept

In AML, the Process-Product-Resource-concept is modelled by means of the CAEX role concept (see IEC 62424:~~2008~~ 2016, A.2.9) and relations between the elements (see 5.5.3). 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).

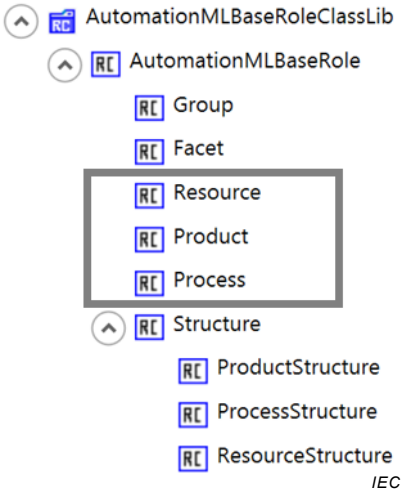


Figure A.29 – AML roles required for the Process-Product-Resource concept

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).

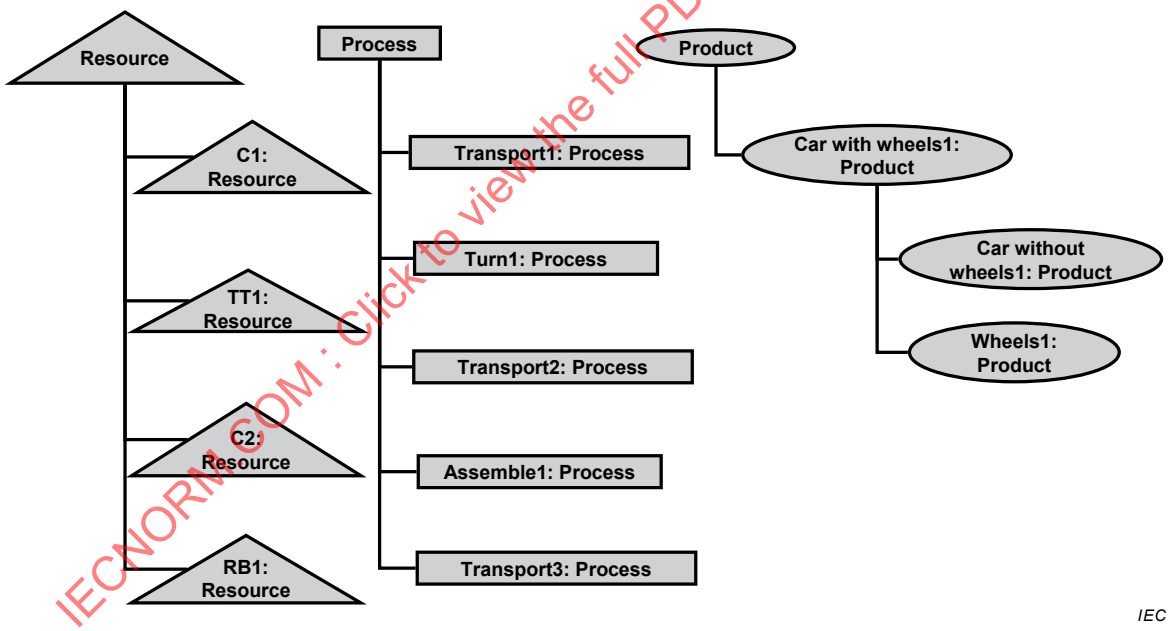


Figure A.30 – Elements of the example

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.

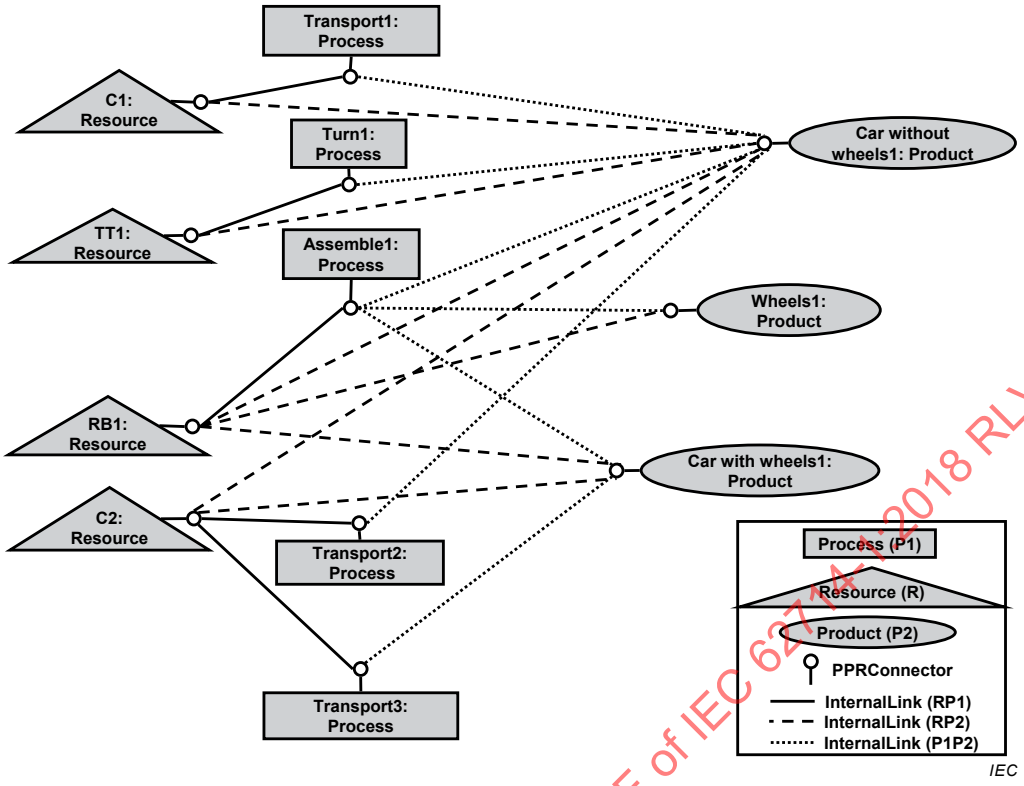


Figure A.31 – Links within the example

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.

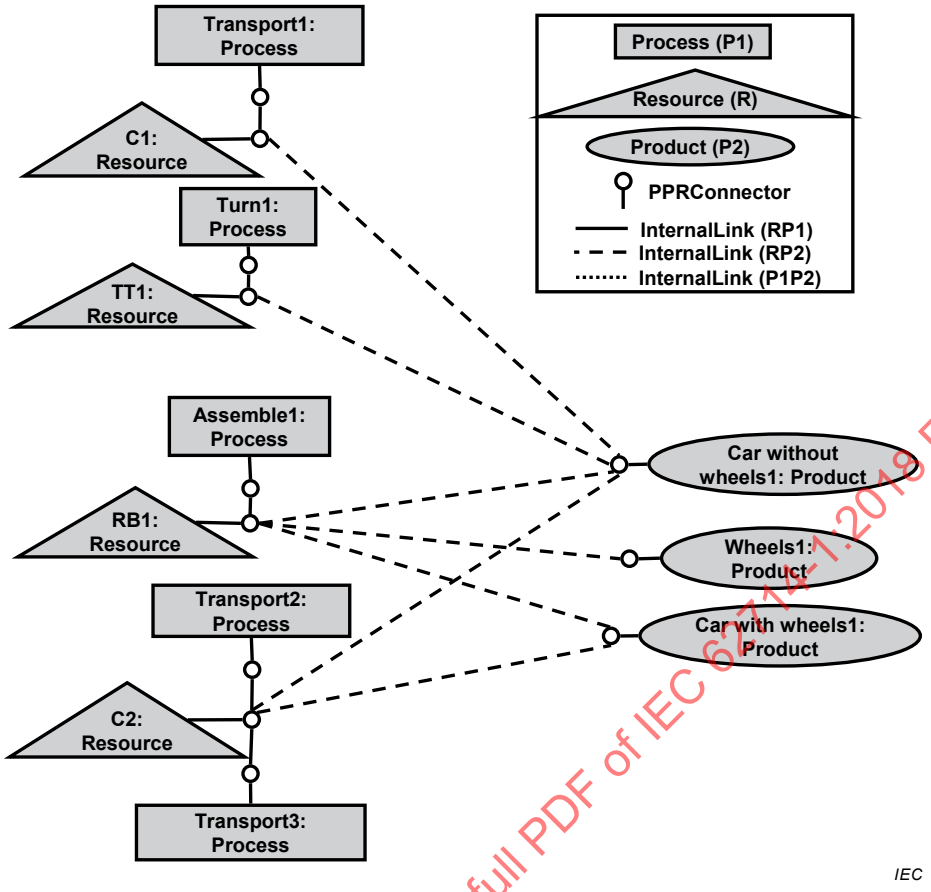


Figure A.32 – Links of the resource centric view on the example

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

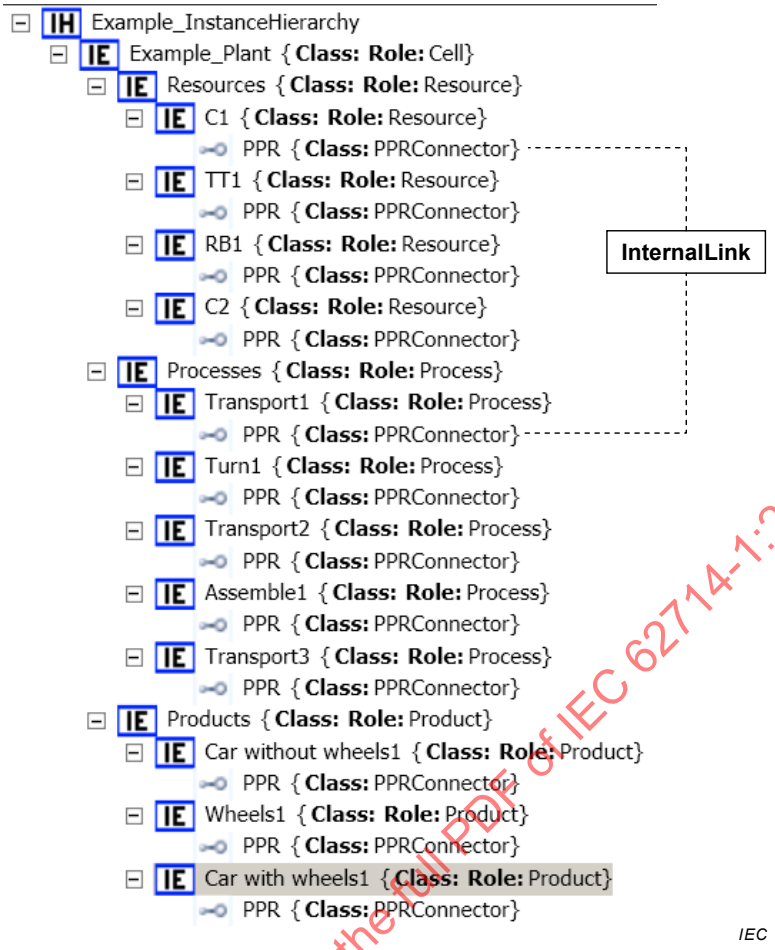


Figure A.33 – InstanceHierarchy of the example in AML

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.

InstanceHierarchy

= Name	Example_InstanceHierarchy

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	CarWithoutWheels1	GUID14
2	Wheels1	GUID15
3	CarWithWheels1	GUID16

IEC

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

IEC

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" ID="GUID3_PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
      </InternalElement>
      <InternalElement Name="TT1" ID="GUID4">
        <ExternalInterface Name="PPR" ID="GUID4_PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
      </InternalElement>
      <InternalElement Name="RB1" ID="GUID5">
        <ExternalInterface Name="PPR" ID="GUID5_PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
      </InternalElement>
      <InternalElement Name="C2" ID="GUID6">
        <ExternalInterface Name="PPR" ID="GUID6_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" ID="GUID8_PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Process"/>
      </InternalElement>
      <InternalElement Name="Turn1" ID="GUID9">
        <ExternalInterface Name="PPR" ID="GUID9_PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Process"/>
      </InternalElement>
      <InternalElement Name="Transport2" ID="GUID10">
        <ExternalInterface Name="PPR" ID="GUID10_PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Process"/>
      </InternalElement>
      <InternalElement Name="Assemble1" ID="GUID11">
        <ExternalInterface Name="PPR" ID="GUID11_PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Process"/>
      </InternalElement>
      <InternalElement Name="Transport3" ID="GUID12">
        <ExternalInterface Name="PPR" ID="GUID12_PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Process"/>
      </InternalElement>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Process"/>
    </InternalElement>
    <InternalElement Name="Products" ID="GUID13">
      <InternalElement Name="CarWithoutWheels1" ID="GUID14">
        <ExternalInterface Name="PPR" ID="GUID14_PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Product"/>
      </InternalElement>
      <InternalElement Name="Wheels1" ID="GUID15">
        <ExternalInterface Name="PPR" ID="GUID15_PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Product"/>
      </InternalElement>
      <InternalElement Name="CarWithWheels1" ID="GUID16">
        <ExternalInterface Name="PPR" ID="GUID16_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>
```

IEC

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.~~

- **InstanceHierarchy**
  - **Name** multipleRolesSupport
  - **InternalElement**
    - **Name** MultiDevice01
    - **ID** GUID1
    - **InternalElement**
      - **Name** D001
      - **ID** GUID2
      - **Attribute** (3)

	Name
1	FaxBoudRate
2	PrintSpeed
3	ScanSpeed

      - **ExternalInterface** (2)

	Name
1	PowerSupply
2	USB

      - **SupportedRoleClass** (3)

	RefRoleClassPath
1	UserdefinedRoleClassLib/Printer
2	UserdefinedRoleClassLib/Fax
3	UserdefinedRoleClassLib/Scanner

      - **RoleRequirements**
        - **RefBaseRoleClassPath** UserdefinedRoleClassLib/Printer
        - **Attribute** (3)

	Name	Value
1	Speed	20
2	Fax.Speed	54
3	Scanner.Speed	1

      - **MappingObject**
        - **AttributeNameMapping** (3)

	RoleAttributeName	SystemUnitAttributeName
1	Speed	PrintSpeed
2	Fax.Speed	FaxBoudRate
3	Scanner.Speed	ScanSpeed

        - **InterfaceNameMapping** (1)

	RoleInterfaceName	SystemUnitInterfaceName
1	Power	PowerSupply

~~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.

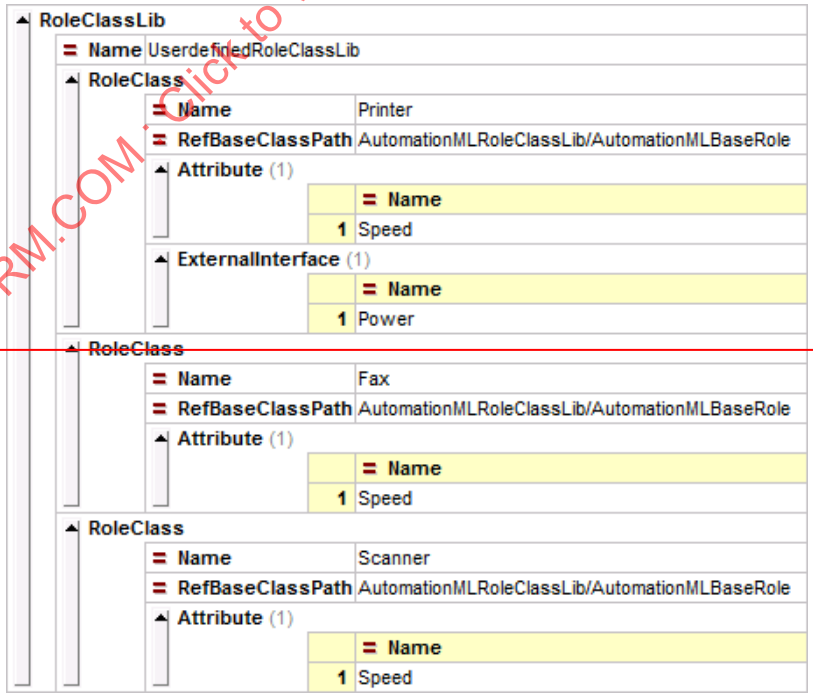


Figure A.39 – AML Role class library corresponding to the multiple role definition example

```
<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**~~

A.2.6 AML multilingual expression concept

A.2.6.1 Concept description

The concept of multilingual expressions aims for storing multi language information within the same AML file.

A.2.6.2 Example of a label attribute with 3 localizations

Figure A.37 gives an example for the AML multilingual expression. The object “PC123” comprises the attribute “Label". The attribute “Label” itself contains the expression in the default language. Modelling localized language texts requires nested attributes. For this purpose, the attribute “Label” comprises the sub-attributes "en-US", "de-DE", and "fr-FR". The names of the sub-attributes are corresponding to language codes according to RFC 5646, their values contain the corresponding localized texts.

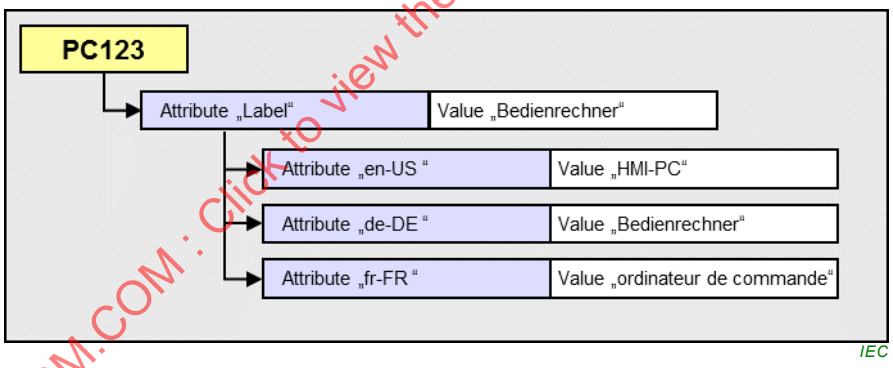


Figure A.37 – Example describing the AML multilingual expression concept

Figure A.38 shows the modelling of the example with AML according to the provisions described in 8.6.

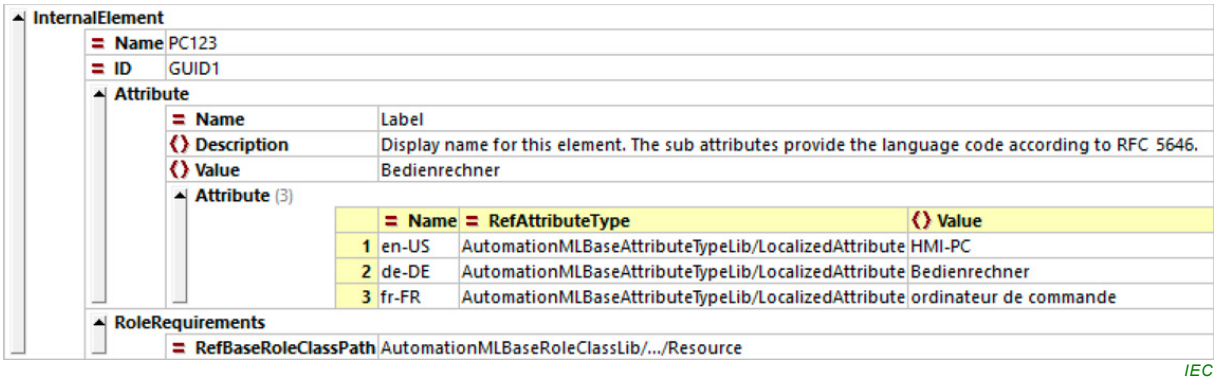


Figure A.38 – XML description of the AML multilingual expression concept

Figure A.39 shows the corresponding XML code of this example.

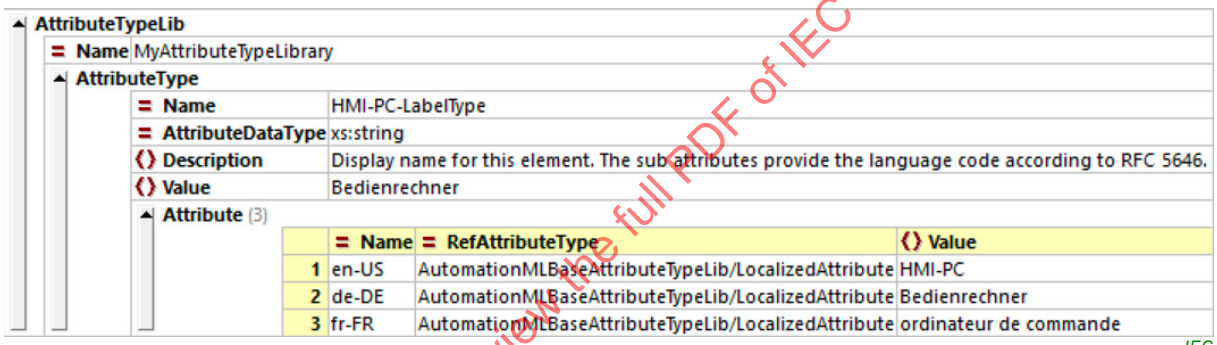
```
<InternalElement Name="PC123" ID="GUID1">
    <Attribute Name="Label">
        <Description>Display name for this element. The sub attributes provide the language code according to RFC 5646.</Description>
        <Value>Bedienrechner</Value>
        <Attribute Name="en-US" RefAttributeType="AutomationMLBaseAttributeTypeLib/LocalizedAttribute">
            <Value>HMI-PC</Value>
        </Attribute>
        <Attribute Name="de-DE" RefAttributeType="AutomationMLBaseAttributeTypeLib/LocalizedAttribute">
            <Value>Bedienrechner</Value>
        </Attribute>
        <Attribute Name="fr-FR" RefAttributeType="AutomationMLBaseAttributeTypeLib/LocalizedAttribute">
            <Value>ordinateur de commande</Value>
        </Attribute>
    </Attribute>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
</InternalElement>
```

IEC

Figure A.39 – XML text describing the AML multilingual expression concept

A.2.6.3 Example of a label attribute type with 3 localizations

Another use case is the predefinition of attributes including all required localized language expressions in a library. Thus, if the label of a specific object, e.g. an HMI-PC, is frequently used, it is useful to model it in a library. Figure A.40 illustrates this by the CAEX AttributeType *HMI-PC-LabelType*. It predefines all 3 localized language expressions.



IEC

Figure A.40 – AML model of a multilingual AttributeType

Figure A.41 shows the related XML code of the given example.

```
<AttributeTypeLib Name="MyAttributeTypeLibrary">
    <AttributeType Name="HMI-PC-LabelType" AttributeDataType="xs:string">
        <Description>Display name for this element. The sub attributes provide the language code according to RFC 5646.</Description>
        <Value>Bedienrechner</Value>
        <Attribute Name="en-US" RefAttributeType="AutomationMLBaseAttributeTypeLib/LocalizedAttribute">
            <Value>HMI-PC</Value>
        </Attribute>
        <Attribute Name="de-DE" RefAttributeType="AutomationMLBaseAttributeTypeLib/LocalizedAttribute">
            <Value>Bedienrechner</Value>
        </Attribute>
        <Attribute Name="fr-FR" RefAttributeType="AutomationMLBaseAttributeTypeLib/LocalizedAttribute">
            <Value>ordinateur de commande</Value>
        </Attribute>
    </AttributeType>
</AttributeTypeLib>
```

IEC

Figure A.41 – XML code of the a multilingual AttributeType

Wherever the label of a HMI-PC is needed including localized language expressions, it needs to reference this attribute type and gets all predefined localized language expressions. This technique can be used to model arbitrary project specific multilingual expressions in a library.

A.2.7 Attribute lists and arrays

A.2.7.1 Concept description

In practice, attributes are often structured. Lists or arrays of data types have to be modelled and exchanged between engineering tools. A list consists of items of the same data type, and may contain further lists, enabling the modelling of arrays of arbitrary dimensions.

Finally, a list is modelled in AML as a *list attribute* containing nested child attributes. The list attribute acts as container for the list. In order to declare a CAEX attribute as a list, it references the AttributeType "ListType" or "OrderedListType", which models a non-ordered list or an ordered list respectively. The child attributes form the list items.

Since CAEX Attributes do not explicitly model the order of the items, the ordered list and the non-ordered list needs to be modelled explicitly.

Normative provisions related to the modelling of lists and arrays are defined in 8.8.

A.2.7.2 Examples

The example in Figure A.42 shows an object *Radio* that stores a list of supported frequencies. The attribute "SupportedFrequencies" references the AML AttributeType "OrderedListType". The child attributes form the list items, each name contains the index of the item within the list. Leading zeros are allowed and exemplarily added for better sortability.

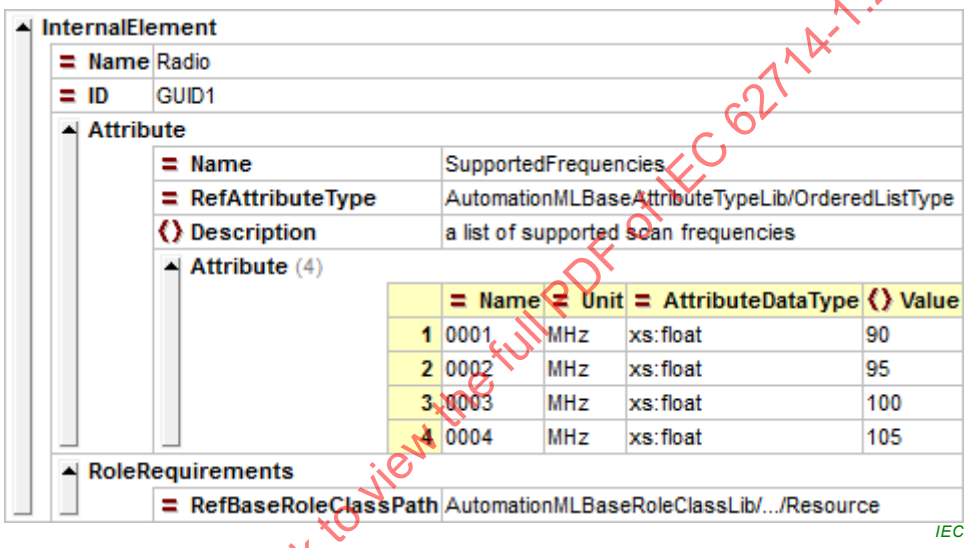


Figure A.42 – Attribute list "SupportedFrequencies"

Figure A.43 shows the related XML code of the example.

```
<InternalElement Name="Radio" ID="GUID1">
  <Attribute Name="SupportedFrequencies" RefAttributeType="AutomationMLBaseAttributeTypeLib/OrderedListType">
    <Description>a list of supported scan frequencies</Description>
    <Attribute Name="0001" Unit="MHz" AttributeDataType="xs:float">
      <Value>90</Value>
    </Attribute>
    <Attribute Name="0002" Unit="MHz" AttributeDataType="xs:float">
      <Value>95</Value>
    </Attribute>
    <Attribute Name="0003" Unit="MHz" AttributeDataType="xs:float">
      <Value>100</Value>
    </Attribute>
    <Attribute Name="0004" Unit="MHz" AttributeDataType="xs:float">
      <Value>105</Value>
    </Attribute>
  </Attribute>
  <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
</InternalElement>
```

IEC

Figure A.43 – XML code for the attribute list "SupportedFrequencies"

The example in Figure A.44 shows the modelling of an array by means of an object "Table" that stores a list of edge points. The attribute "Edges" references the AML AttributeType

"ListType". The child attributes form the list items, the names are arbitrary but unique among the siblings. They form again lists, containing the x, y and z positions of the table.

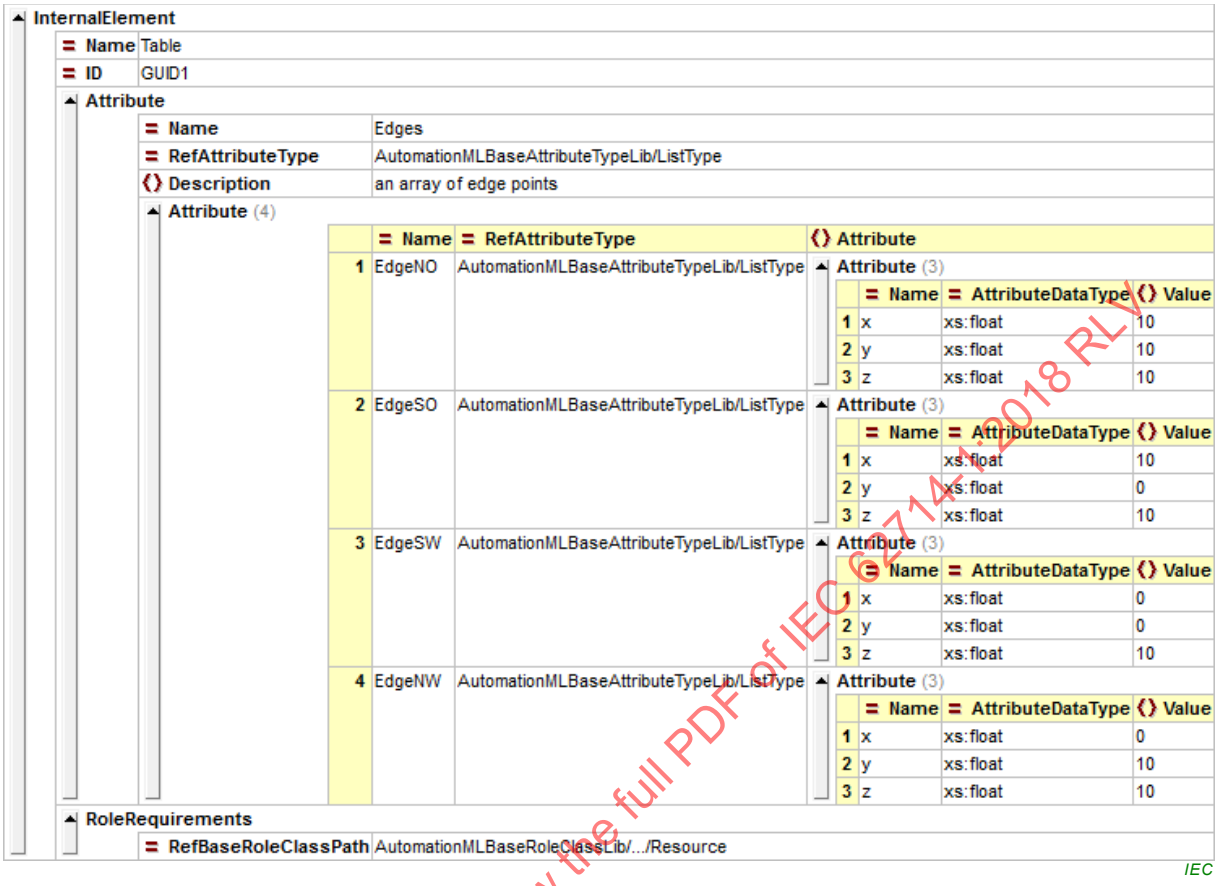


Figure A.44 – Example CAEX model of the array "Edges"

Figure A.45 shows the related XML code of the example.

```
<InternalElement Name="Table" ID="GUID1">
  <Attribute Name="Edges" RefAttributeType="AutomationMLBaseAttributeTypeLib/ListType">
    <Description>an array of edge points</Description>
    <Attribute Name="EdgeNO" RefAttributeType="AutomationMLBaseAttributeTypeLib/ListType">
      <Attribute Name="x" AttributeDataType="xs:float">
        <Value>10</Value>
      </Attribute>
      <Attribute Name="y" AttributeDataType="xs:float">
        <Value>10</Value>
      </Attribute>
      <Attribute Name="z" AttributeDataType="xs:float">
        <Value>10</Value>
      </Attribute>
    </Attribute>
    <Attribute Name="EdgeSO" RefAttributeType="AutomationMLBaseAttributeTypeLib/ListType">
      <Attribute Name="x" AttributeDataType="xs:float">
        <Value>10</Value>
      </Attribute>
      <Attribute Name="y" AttributeDataType="xs:float">
        <Value>0</Value>
      </Attribute>
      <Attribute Name="z" AttributeDataType="xs:float">
        <Value>10</Value>
      </Attribute>
    </Attribute>
    <Attribute Name="EdgeSW" RefAttributeType="AutomationMLBaseAttributeTypeLib/ListType">
      <Attribute Name="x" AttributeDataType="xs:float">
        <Value>0</Value>
      </Attribute>
      <Attribute Name="y" AttributeDataType="xs:float">
        <Value>0</Value>
      </Attribute>
      <Attribute Name="z" AttributeDataType="xs:float">
        <Value>10</Value>
      </Attribute>
    </Attribute>
    <Attribute Name="EdgeNW" RefAttributeType="AutomationMLBaseAttributeTypeLib/ListType">
      <Attribute Name="x" AttributeDataType="xs:float">
        <Value>0</Value>
      </Attribute>
      <Attribute Name="y" AttributeDataType="xs:float">
        <Value>10</Value>
      </Attribute>
      <Attribute Name="z" AttributeDataType="xs:float">
        <Value>10</Value>
      </Attribute>
    </Attribute>
  </Attribute>
  <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
</InternalElement>
```

IEC

Figure B.45 – XML code for the attribute array "Edges"

Annex B
(informative)

XML representation of standard AML base 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="9942bd9e-e19d-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="OPCUADAInterface"
    RefBaseClassPath="ExternalDataConnector" />
    <InterfaceClass Name="PLCopenXMLInterface"
    RefBaseClassPath="ExternalDataConnector" />
   </InterfaceClass>
   <InterfaceClass Name="Communication"
    RefBaseClassPath="AutomationMLBaseInterface">
    <InterfaceClass Name="SignalInterface" RefBaseClassPath="Communication"
     />
    </InterfaceClass>
   </InterfaceClass>
 </InterfaceClassLib>
</CAEXFile>
```

Figure B.1 below shows the XML text of the standard AML interface class library, role class library and attribute type library.

```
<CAEXFile xmlns="http://www.dke.de/CAEX" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
          instance" SchemaVersion="3.0" FileName="AutomationML2.10BaseLibraries"
          xsi:schemaLocation="http://www.dke.de/CAEX CAEX_ClassModel_V.3.0.xsd">
 <SuperiorStandardVersion>AutomationML 2.10</SuperiorStandardVersion>
 <SourceDocumentInformation OriginID="IEC SC65E WG 9" OriginName="IEC SC65E WG 9"
          OriginVersion="2.10.0" LastWritingDateTime="2016-08-25T09:58:00.0Z"
          OriginProjectID="Automation Markup Language Standard Library"
          OriginRelease="2.10.0" OriginVendor="IEC" OriginVendorURL="www.iec.ch"
          OriginProjectTitle="Automation Markup Language Standard Libraries"/>
 <InterfaceClassLib Name="AutomationMLInterfaceClassLib">
  <Description>Standard Automation Markup Language Interface Class
          Library</Description>
  <Version>2.10.0</Version>
  <InterfaceClass Name="AutomationMLBaseInterface">
   <InterfaceClass Name="Order" RefBaseClassPath="AutomationMLBaseInterface">
      <Attribute Name="Direction" AttributeDataType="xs:string"
         RefAttributeType="AutomationMLBaseAttributeTypeLib/Direction"/>
   </InterfaceClass>
   <InterfaceClass Name="Port" RefBaseClassPath="AutomationMLBaseInterface">
      <Attribute Name="Direction" AttributeDataType="xs:string"
         RefAttributeType="AutomationMLBaseAttributeTypeLib/Direction">
       <Constraint Name="AllowedValues">
        <NominalScaledType>
        <RequiredValue>In</RequiredValue>
        <RequiredValue>Out</RequiredValue>
        <RequiredValue>InOut</RequiredValue>
        </NominalScaledType>
       </Constraint>
      </Attribute>
      <Attribute Name="Cardinality"
         RefAttributeType="AutomationMLBaseAttributeTypeLib/Cardinality">
       <Attribute Name="MinOccur" AttributeDataType="xs:unsignedInt"/>
       <Attribute Name="MaxOccur" AttributeDataType="xs:unsignedInt"/>
      </Attribute>
      <Attribute Name="Category" AttributeDataType="xs:string"
         RefAttributeType="AutomationMLBaseAttributeTypeLib/Category"/>
   </InterfaceClass>
   <InterfaceClass Name="PPRConnector" RefBaseClassPath="AutomationMLBaseInterface"/>
   <InterfaceClass Name="ExternalDataConnector"
          RefBaseClassPath="AutomationMLBaseInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI"
         RefAttributeType="AutomationMLBaseAttributeTypeLib/refURI"/>
      <InterfaceClass Name="COLLADAInterface"
         RefBaseClassPath="ExternalDataConnector"/>
      <InterfaceClass Name="PLCopenXMLInterface"
         RefBaseClassPath="ExternalDataConnector"/>
      <InterfaceClass Name="ExternalDataReference"
         RefBaseClassPath="ExternalDataConnector">
       <Attribute Name="MIMEType" AttributeDataType="xs:string"
         RefAttributeType="AutomationMLBaseAttributeTypeLib/MIMEType"/>
      </InterfaceClass>
   </InterfaceClass>
   <InterfaceClass Name="Communication" RefBaseClassPath="AutomationMLBaseInterface">
      <InterfaceClass Name="SignalInterface" RefBaseClassPath="Communication"/>
   </InterfaceClass>
  </InterfaceClass>
 </InterfaceClassLib>
 <RoleClassLib Name="AutomationMLBaseRoleClassLib">
  <Description>Automation Markup Language base role class library</Description>
  <Version>2.10.0</Version>
  <RoleClass Name="AutomationMLBaseRole">
   <RoleClass Name="Group" RefBaseClassPath="AutomationMLBaseRole">
      <Attribute Name="AssociatedFacet" AttributeDataType="xs:string"
         RefAttributeType="AutomationMLBaseAttributeTypeLib/AssociatedFacet"/>
```

```
    </RoleClass>
    <RoleClass Name="Facet" RefBaseClassPath="AutomationMLBaseRole"/>
    <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="ExternalData" RefBaseClassPath="AutomationMLBaseRole"/>
  </RoleClass>
 </RoleClassLib>
 <AttributeTypeLib Name="AutomationMLBaseAttributeTypeLib">
  <Description>Standard Automation Markup Language Attribute Type Library</Description>
  <Version>2.10.0</Version>
  <AttributeType Name="Direction" AttributeDataType="xs:string">
   <Constraint Name="AllowedValues">
      <NominalScaledType>
       <RequiredValue>In</RequiredValue>
       <RequiredValue>Out</RequiredValue>
       <RequiredValue>InOut</RequiredValue>
      </NominalScaledType>
   </Constraint>
  </AttributeType>
  <AttributeType Name="Cardinality">
   <Attribute Name="MinOccur" AttributeDataType="xs:unsignedInt"/>
   <Attribute Name="MaxOccur" AttributeDataType="xs:unsignedInt"/>
  </AttributeType>
  <AttributeType Name="Category" AttributeDataType="xs:string"/>
  <AttributeType Name="refURI" AttributeDataType="xs:anyURI"/>
  <AttributeType Name="AssociatedFacet" AttributeDataType="xs:string"/>
  <AttributeType Name="ListType"/>
  <AttributeType Name="OrderedListType"/>
  <AttributeType Name="LocalizedAttribute" AttributeDataType="xs:string"/>
  <AttributeType Name="AssociatedExternalValue">
   <Attribute Name="refCAEXAttribute"/>
   <Attribute Name="refURI"
         RefAttributeType="AutomationMLBaseAttributeTypeLib/refURI"/>
   <Attribute Name="Direction"
         RefAttributeType="AutomationMLBaseAttributeTypeLib/Direction"/>
  </AttributeType>
  <AttributeType Name="MIMEType" AttributeDataType="xs:string"/>
  <AttributeType Name="DocLang" AttributeDataType="xs:string"/>
 </AttributeTypeLib>
</CAEXFile>
```

Figure B.1 – XML text of the standard AML interface class library, role class library and attribute type library

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*[3]

IEC 62714-3, *Engineering data exchange format for use in industrial automation systems engineering – Automation markup language – Part 3: Geometry and kinematics*

IEC 62714-4:__[4], *Engineering data exchange format for use in industrial automation systems engineering – Automation Markup Language – Part 4: Logic*

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

Extensible Markup Language (XML) 1.0 1.0:2004, W3C Recommendation [viewed 2017-11-13]. Available at http://www.w3.org/TR/2004/REC-xml-20040204/

[3] To be published.

[4] Under consideration.

IEC 62714-1

Edition 2.0 2018-04

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 62714-1:2018-04(en-fr)

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.

This second edition cancels and replaces the first edition published in 2014. This edition constitutes a technical revision.

This edition includes the following significant technical changes with respect to the previous edition:

a) use of CAEX 3.0 according to IEC 62424:2016 which provides technical improvements as attribute libraries, nested interfaces, new fields for indicating the source of an object, a refinement of the mirror concept and native support of multiple roles, native meta information about the CAEX file source tool, identification of instances via unique IDs instead of pathes, etc.,

b) improved modelling of references to documents outside of the scope of the present standard,

c) modelling of references between CAEX attributes and items in external documents, e.g. within an Excel sheet,

d) revised role libraries,

e) modified Port concept,

f) modelling of multilingual expressions,

g) modelling of structured attribute lists or array,

h) a new AML container format,

i) a new standard AML attribute library.

The text of this International Standard is based on the following documents:

FDIS	Report on voting
65E/582/FDIS	65E/586/RVD

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

This document 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 document will remain unchanged until the stability date indicated on the IEC website under "http://webstore.iec.ch" in the data related to the specific document. At this date, the document 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 for plant engineering data. AML 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. The application of IEC 62714 is industry independent. It is applicable in all industries that require data exchange in their engineering tool chain, e.g. in discrete industry or process industry.

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 can 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. CAEX is utilized to interconnect 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.

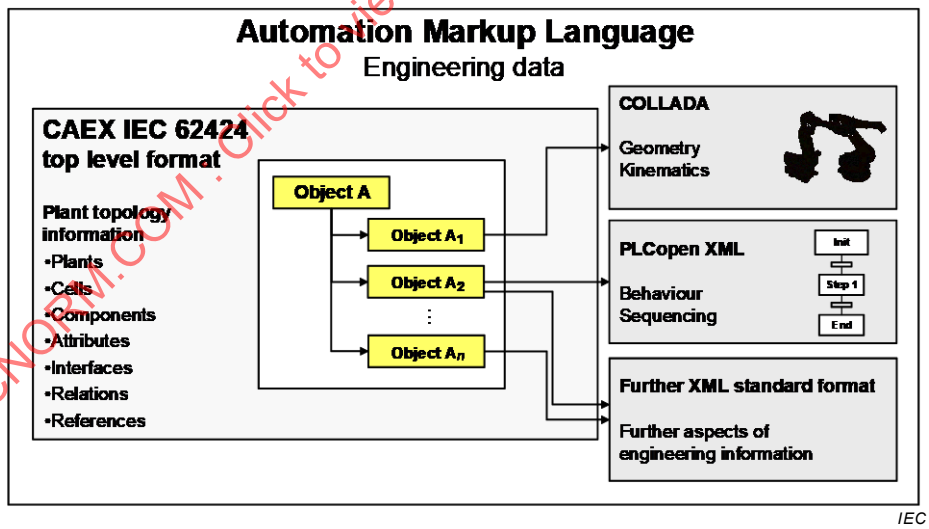


Figure 1 – Overview of the engineering data exchange format AML

Due to the different aspects of AML, the IEC 62714 series consists of different parts focusing 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 subformats.

- IEC 62714-2: Role class libraries

  This part specifies additional AML libraries.

- IEC 62714-3: Geometry and kinematics

  This part specifies the modelling of geometry and kinematics information.

- IEC 62714-4[1]: Logic

  This part specifies the modelling of logics, sequencing, behaviour and control related information.

Further parts will 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.

Clause 1 defines the scope for IEC 62714.

Clause 2 provides normative references.

Clause 3 provides terms, definitions and abbreviations.

Clause 4 defines the conformity to IEC 62714.

Clause 5 describes general architecture specifications for IEC 62714.

Clause 6 defines the basic AML libraries.

Clause 7 describes how to model user-defined data.

Clause 8 describes extended AML concepts.

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.

[1] Under consideration.

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 automation markup language (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 are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 62424:2016, *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/PAS 17506, *Industrial automation systems and integration – COLLADA digital asset schema specification for 3D visualization of industrial data*

ISO/IEC 29500-2, *Information technology – Document description and processing languages – Office Open XML File Formats – Part 2: Open Packaging Conventions*

IETF RFC 2046, *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types* [viewed 2017-11-13]. Available at <http://www.ietf.org>

IETF RFC 4122, A Universally Unique Identifier (UUID) URN Namespace
[viewed 2017-11-13]. Available at <http://www.ietf.org>

IETF RFC 5646, *Tags for Identifying Languages*
[viewed 2017-11-13]. Available at <http://www.ietf.org>

COLLADA 1.4.1:March 2008, COLLADA – Digital Asset Schema Release 1.4.1
[viewed 2017-11-13]. Available at <http://www.khronos.org/files/collada_spec_1_4.pdf>

PLCopen XML 2.0:December 3rd 2008 and PLCopen XML 2.0.1:May 8th 2009, *XML formats for IEC 61131-3* [viewed 2017-11-13]. 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.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at http://www.electropedia.org/
- ISO Online browsing platform: available at http://www.iso.org/obp

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 one or more CAEX RoleRequirements that relate 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, either an AML system unit class, AML interface class, AML role class or AML attribute type

Note 1 to entry: AML classes are stored within AML libraries, AML classes are of type SystemUnitClass, InterfaceClass, RoleClass or AttributeType.

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.

Note 4 to entry: AML classes can be user-defined or standard AML classes.

3.1.5
AML attribute
CAEX attribute which belongs to an AML object and is related to an attribute defined in an AML class or AML AttributeType

Note 1 to entry: AML attributes are described as an XML element corresponding to IEC 62424:2016, A.2.4.

3.1.6
AML document
certain AML CAEX document following IEC 62714 (all parts) 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.

Note 2 to entry: AML documents contain AML objects and/or user-defined objects.

Note 3 to entry: An AML document may consist of multiple files, with one AML CAEX document as root.

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

3.1.8
AML interface
single connection point with a relation to an AML interface class

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 interface with a direct or indirect relation to the standard AML interface class Port, allowing to specify nested interfaces

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 with a direct or indirect relation to the standard AML role class Group, providing a certain view on AML objects

3.1.12
AML Facet
AML object with a direct or indirect relation to the standard AML role class Facet, 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 62424:2016, 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 automatically 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

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 (all parts), GUIDs are also presented in a short form such as “GUID1”, “GUID2”, etc. This serves the readability and acts as a real GUID.

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
topology
hierarchical structure of a system, visualizable as object tree

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

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

3.1.21
publish
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.

3.1.22
relation
association between CAEX objects

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

3.1.23
link
connection between objects of type CAEX ExternalInterface

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

3.1.24
reference
association between a CAEX InternalElement and externally stored information

3.2 Abbreviations

The abbreviations used in this document are listed in Table 1.

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 62424:2016, 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 focusing 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 62424:2016, 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 62424:2016, A.2.8.7.

NOTE 1 According to IEC 62424:2016, A.2.8.7, 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 (all parts) 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 62424:2016, Annex D.

Reference and relation information: References and relations shall be stored according to 5.5 and 5.6. 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.1.7. 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™[2]. COLLADA interfaces that need to be interconnected within the top level format shall be published as CAEX ExternalInterfaces.

NOTE 5 IEC 62714 (all parts) does not syntactically modify the COLLADA data format. An overview example of how to reference COLLADA is provided in A.1.3. Details are 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.

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 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 Sub document versions and AML superior document information

IEC 62714 is based on the following document formats:

- CAEX version 3.0 as defined in IEC 62424:2016;
- 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.

AML integrates CAEX, hence AML acts as superior standard.

NOTE 1 Normative provisions regarding the version information related to AML object instances are defined in 8.7. The storage of tool specific meta information is defined in 5.4.

Hence, the following provisions apply:

- Each AML CAEX document shall store the AML version which this standard follows in the CAEX element “SuperiorStandardVersion” according to IEC 62424:2016, A.2.2.3.

---

2 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.

- The value of this element shall be “AutomationML 2.10” in order to correspond to the present 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 2.10.

```
<CAEXFile xmlns="http://www.dke.de/CAEX" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
SchemaVersion="3.0" FileName="AutomationML2.10BaseLibraries" xsi:schemaLocation="
http://www.dke.de/CAEX CAEX_ClassModel_V.3.0.xsd">
  <SuperiorStandardVersion>AutomationML 2.10</SuperiorStandardVersion>
```

IEC

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.

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:

- According to IEC 62424:2016, each AML document shall provide information about the source 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 can hinder the iterative data exchange with the other tools: hence the removal of writer information of other tools is not recommended.
- This document recommends to use the optional CAEX element SourceObjectInformation with its attributes OriginID and SourceObjID according to IEC 62424:2016, A.2.2.7, in order to identify the source tool of each AML Object instance (InternalElement, ExternalInterface).

Figure 3 illustrates the required XML text of the required document origin information. The example shows the source information of the standard libraries provided with this part of IEC 62714.

```
  <SourceDocumentInformation OriginID="IEC SC65E WG 9" OriginName="IEC SC65E WG 9"
OriginVersion="2.10.0" LastWritingDateTime="2016-08-25T09:58:00.0Z" OriginProjectID="Automation
Markup Language Standard Library" OriginRelease="2.10.0" OriginVendor="IEC" OriginVendorURL="
www.iec.ch" OriginProjectTitle="Automation Markup Language Standard Libraries"/>
```

IEC

Figure 3 – XML text of the AML source tool information

5.5 AML relations specification

5.5.1 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 5.5 focuses on relations, whereas 5.6 focuses on references. An informative overview about relations and references is provided in A.1.7.

5.5.2 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 a CAEX InstanceHierarchy or of a CAEX 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.
- According to IEC 62424:2016, A.2.2.7, 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”. Additionally, within the old class, the path to the new version should be stored in the CAEX tag “NewVersion”.

  NOTE 3 This provision supports tracking of changes across different versions of a class.

5.5.3 Instance-instance-relations

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

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

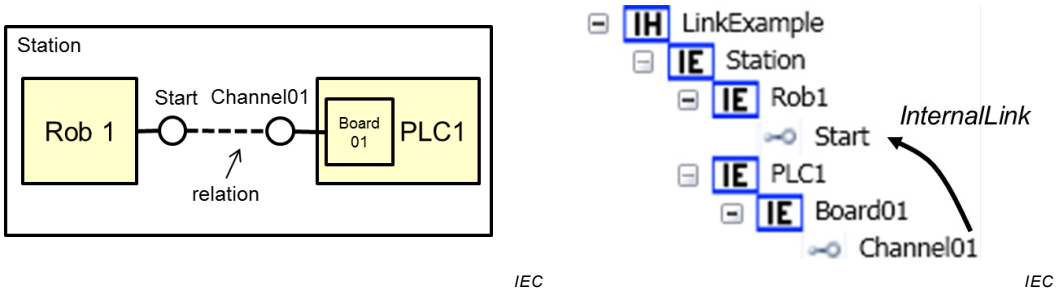
- The ExternalInterfaces should be derived directly or indirectly from an AML standard interface class.

  NOTE 1 The AML standard interface class library is specified in 6.3. The AML interface class defines 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 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 specified in IEC 62714-4.

Figure 4a) describes an example comprising a robot “Rob1” and a PLC “PLC1”, each with one signal interface that are connected. Figure 4b) depicts this example as an object hierarchy.



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

Figure 4 – Example of a relation as block diagram and as object tree

Figure 5 and Figure 6 depict the AML representation of the given example. The full XML text for the InstanceHierarchy for this example comprising all InternalElements “Station”, “Rob1”, “PLC1” and “Board01” including their interfaces is shown below.

NOTE 4 The path strings given in this example (see Figure 4) are reduced with “/.../” in order to increase the readability.

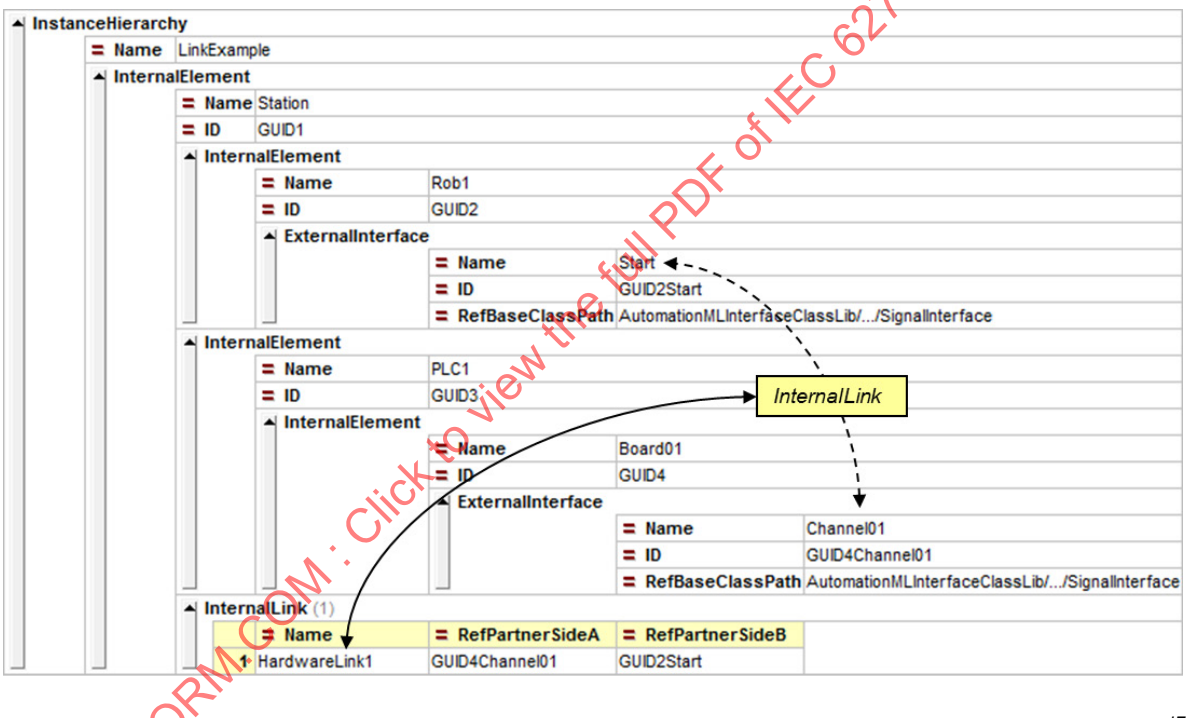


Figure 5 – Example relation between the objects “PLC1” and “Rob1”

```
<InstanceHierarchy Name="LinkExample">
  <InternalElement Name="Station" ID="GUID1">
    <InternalElement Name="Rob1" ID="GUID2">
      <ExternalInterface Name="Start" ID="GUID2Start" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface"/>
    </InternalElement>
    <InternalElement Name="PLC1" ID="GUID3">
      <InternalElement Name="Board01" ID="GUID4">
        <ExternalInterface Name="Channel01" ID="GUID4Channel01" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface"/>
      </InternalElement>
    </InternalElement>
    <InternalLink Name="HardwareLink1" RefPartnerSideA="GUID4Channel01" RefPartnerSideB="GUID2Start"/>
  </InternalElement>
</InstanceHierarchy>
```

IEC

Figure 6 – XML text of the example relation between the objects “PLC1” and “Rob1”

5.5.4 Identification of objects

This document recommends the identification of InternalElements and ExternalInterfaces by means of GUIDs according to RFC 4122. For the comparison of two identifiers, the following provision applies:

- If the identifier of an InternalElement or an ExternalInterface is a GUID, then two identifiers shall be identical, when the contained numerical value is identical. Brackets, braces, dashes or spaces are allowed but shall be irrelevant.

  Example without braces: 48d23207-09e0-4104-82fb-344007d2b7f5

  Example with braces: { 48d23207-09e0-4104-82fb-344007d2b7f5 }

5.6 AML document reference specification

5.6.1 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.

5.6.2 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 specified in IEC 62714-3.

5.6.3 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 specified in IEC 62714-4.

5.6.4 Referencing additional documents in the scope of IEC 62714 (all parts)

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 part of the series. 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.

5.6.5 Referencing documents outside of the scope of IEC 62714 (all parts)

If an external document, which is not in the scope of this standard, needs to be referenced by the AML document (e.g. manuals, instructions or specific engineering results like native control programs) the following provisions apply:

- A document which is outside of the scope of IEC 62714 shall be modelled by means of a CAEX InternalElement with a direct or indirect association to the RoleClass “ExternalData” which is defined in 6.4.12. The referenced RoleClass shall specify the content of the document. More than one role with content types can be assigned to a document.

  NOTE 1 Each document can contain contents of several types, e.g. bill of material and user manual.

  NOTE 2 Each document can reference to several files if necessary, e.g. if it splitted in different files.

- If a document is language specific, it shall contain a CAEX attribute of type “DocLang”. If a document contains more than one language, it shall contain an unsorted attribute list as described in 8.8 with attributes of type “DocLang”.
- Each document shall contain one or more ExternalInterfaces which shall be directly or indirectly derived from the interface class “ExternalDataReference”.
- This ExternalInterface shall model the URI to the external document by means of the predefined CAEX attribute of type “refURI” which is inherited from the AML standard interface class “ExternalDataConnector”, and shall additionally model the type of the document by means of the predefined CAEX attribute “MIMEType” of type “MIMEType” which is inherited from the AML standard interface class “ExternalDataReference”.

Additional information and an example are provided in A.1.5.

5.6.6 Referencing CAEX attributes to items in external documents

If a CAEX attribute needs to be associated with a related item in an external document (e.g. an external XML document or an Excel document which is outside of the scope of this standard), the following provisions apply:

- Each reference between a CAEX attribute and an item in an external document shall be modelled by a CAEX ExternalInterface according to the provisions specified in 5.6.5.
- For each reference between a CAEX attribute and an item in an external document, this CAEX ExternalInterface models one additional attribute of type “AssociatedExternalValue” which has nested attributes.
- The first nested attribute shall mirror the CAEX attribute. This means, that the GUID of the attributes parent object, separated by “/” and the name of the attribute is modelled in the CAEX attribute “RefAttributeType”. The name of this attribute shall be irrelevant, but unique among its siblings.
- The second nested attribute of type “refURI” shall reference the item in the external document. This reference shall be the same document or a sub-document of the external document that is referenced in the parent InternalElement defined as described in 5.6.5. The syntax of this reference is outside of the scope of this standard and requires a referencable external document element.
- The third nested attribute of type “Direction” shall model the direction of the information flow. The attribute value shall be “In” if the external attribute is consumed by the CAEX attribute (then the external item is leading), or the attribute value shall be “Out” if the CAEX attribute is leading and the external item consumes the value of the CAEX attribute. The value “InOut” is forbidden.

Additional information and an example are provided in A.1.6.

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 after the instantiation 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 standard role class “AutomationMLBaseRole”.

- All AML interfaces shall be associated directly or indirectly to the standard interface class "AutomationMLBaseInterface".

6.3 AML interface class library – AutomationMLInterfaceClassLib

6.3.1 General

The following "AutomationMLInterfaceClassLib" is modelled according to IEC 62424:2016, Clause 7, Annex A and Annex C. IEC 62714 (all parts) utilizes the CAEX interface concept. User-defined extensions of this AML library are allowed as specified in 7.4.

Each interface shall be derived directly or indirectly from a class of the following standard "AutomationMLInterfaceClassLib" according to Table 2. Subclauses 6.3.2 to 6.3.11 specify the interface classes in detail.

Table 2 – Interface classes of the AutomationMLInterfaceClassLib

AML InterfaceClass library	InterfaceClass	Description
AutomationMLInterfaceClassLib AutomationMLBaseInterface Order Port PPRConnector ExternalDataConnector COLLADAInterface PLCopenXMLInterface ExternalDataReference Communication SignalInterface	AutomationMLBaseInterface	Abstract interface type
	Order	Interface for describing orders
	Port	Interface for describing and interconnecting 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
	ExternalDataReference	Interface to external documents outside of the scope of this standard
	Communication	Generic communication interface
	SignalInterface	Generic signal interface

Figure 7 shows a table view and Figure 8 the XML text of the standard AML InterfaceClassLib. Subclauses 6.3.2 to 6.3.10 provide detail information about the classes.

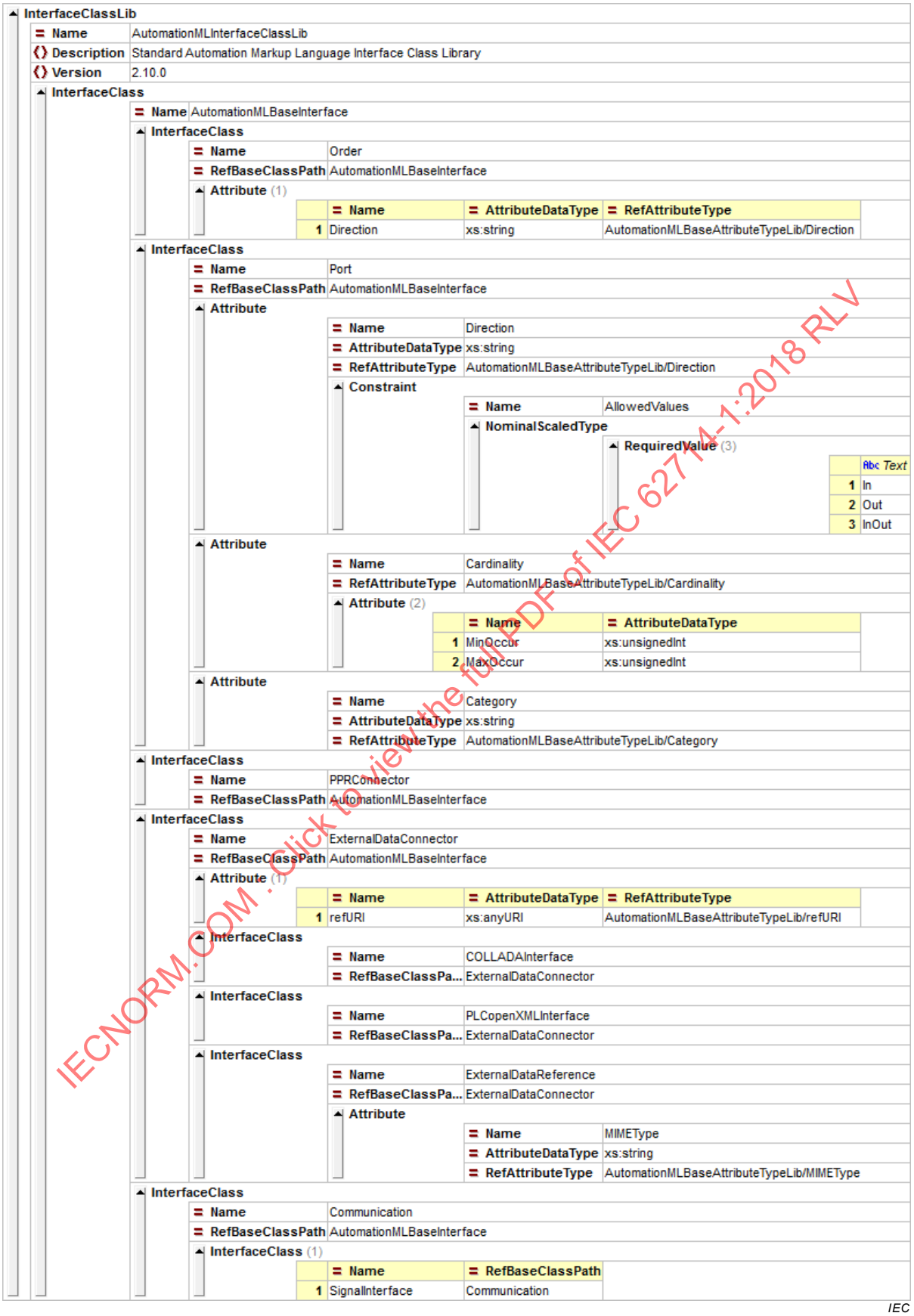


Figure 7 – AML basic interface class library

```
<InterfaceClassLib Name="AutomationMLInterfaceClassLib">
  <Description>Standard Automation Markup Language Interface Class Library</Description>
  <Version>2.10.0</Version>
  <InterfaceClass Name="AutomationMLBaseInterface">
    <InterfaceClass Name="Order" RefBaseClassPath="AutomationMLBaseInterface">
      <Attribute Name="Direction" AttributeDataType="xs:string" RefAttributeType="AutomationMLBaseAttributeTypeLib/Direction"/>
    </InterfaceClass>
    <InterfaceClass Name="Port" RefBaseClassPath="AutomationMLBaseInterface">
      <Attribute Name="Direction" AttributeDataType="xs:string" RefAttributeType="AutomationMLBaseAttributeTypeLib/Direction">
        <Constraint Name="AllowedValues">
          <NominalScaledType>
            <RequiredValue>In</RequiredValue>
            <RequiredValue>Out</RequiredValue>
            <RequiredValue>InOut</RequiredValue>
          </NominalScaledType>
        </Constraint>
      </Attribute>
      <Attribute Name="Cardinality" RefAttributeType="AutomationMLBaseAttributeTypeLib/Cardinality">
        <Attribute Name="MinOccur" AttributeDataType="xs:unsignedInt"/>
        <Attribute Name="MaxOccur" AttributeDataType="xs:unsignedInt"/>
      </Attribute>
      <Attribute Name="Category" AttributeDataType="xs:string" RefAttributeType="AutomationMLBaseAttributeTypeLib/Category"/>
    </InterfaceClass>
    <InterfaceClass Name="PPRConnector" RefBaseClassPath="AutomationMLBaseInterface"/>
    <InterfaceClass Name="ExternalDataConnector" RefBaseClassPath="AutomationMLBaseInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI" RefAttributeType="AutomationMLBaseAttributeTypeLib/refURI"/>
      <InterfaceClass Name="COLLADAInterface" RefBaseClassPath="ExternalDataConnector"/>
      <InterfaceClass Name="PLCopenXMLInterface" RefBaseClassPath="ExternalDataConnector"/>
      <InterfaceClass Name="ExternalDataReference" RefBaseClassPath="ExternalDataConnector">
        <Attribute Name="MIMEType" AttributeDataType="xs:string" RefAttributeType="AutomationMLBaseAttributeTypeLib/MIMEType"/>
      </InterfaceClass>
    </InterfaceClass>
    <InterfaceClass Name="Communication" RefBaseClassPath="AutomationMLBaseInterface">
      <InterfaceClass Name="SignalInterface" RefBaseClassPath="Communication"/>
    </InterfaceClass>
  </InterfaceClass>
</InterfaceClassLib>
```

IEC

Figure 8 – XML description of the AML basic interface class library

6.3.2 InterfaceClass AutomationMLBaseInterface

Table 3 specifies the interface class "AutomationMLBaseInterface".

Table 3 – 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
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface
Attributes	None

6.3.3 InterfaceClass Order

Table 4 specifies the interface class "Order".

Table 4 – 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
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Order
Attributes	Name: Direction RefAttributeType: AutomationMLBaseAttributeTypeLib/Direction Semantics: see 6.5.2

6.3.4 InterfaceClass Port

Table 5 specifies the role class “Port”.

Table 5 – Optional attributes for AML Port interfaces

Class name	Port
Description	The interface class “Port” is an interface type for interfaces that contain a number of nested interfaces and allows describing complex interfaces in this way. AML Port interfaces shall reference this interface class. Details and examples are specified in 8.2.
Parent class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Port
Attributes	Name: Direction RefAttributeType: AutomationMLBaseAttributeTypeLib/Direction Semantics: see 6.5.2
	Name: Cardinality RefAttributeType: AutomationMLBaseAttributeTypeLib/Cardinality Semantics: see 6.5.2
	Name: Category RefAttributeType: AutomationMLBaseAttributeTypeLib/Category Semantics: see 6.5.2

6.3.5 InterfaceClass PPRConnector

Table 6 specifies the interface class “PPRConnector”.

Table 6 – 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.2.5 for more information.
Parent class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/PPRConnector
Attributes	

6.3.6 InterfaceClass ExternalDataConnector

Table 7 specifies the interface class “ExternalDataConnector”.

Table 7 – 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
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector
Attributes	Name: refURI RefAttributeType: AutomationMLBaseAttributeTypeLib/refURI Semantics: see 6.5.2

6.3.7 InterfaceClass COLLADAInterface

Table 8 specifies the interface class “COLLADAInterface”. Details are specified in IEC 62714-3.

Table 8 – 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 specified in IEC 62714-3.
Parent class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ ExternalDataConnector/COLLADAInterface
Attributes	None

6.3.8 InterfaceClass PLCopenXMLInterface

Table 9 specifies the interface class “PLCopenXMLInterface”. Details are specified in IEC 62714-4.

Table 9 – 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 specified in IEC 62714-4.
Parent class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ ExternalDataConnector/PLCopenXMLInterface
Attributes	None

6.3.9 InterfaceClass ExternalDataReference

Table 10 specifies the interface class “ExternalDataReference”. Details are specified in 5.6.5.

Table 10 – InterfaceClass ExternalDataReference

Class name	ExternalDataReference
Description	The interface class “ExternalDataReference” shall be used in order to reference external documents out of the scope of AML. Details are specified in 5.6.5.
Parent class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ ExternalDataConnector/ExternalDataReference
Attributes	Name: MIMEType RefAttributeType: AutomationMLBaseAttributeTypeLib/MIMEType Semantics: see 6.5.2

6.3.10 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
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication
Attributes	None

6.3.11 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 4.
Parent class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication
Path for element reference	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication/SignalInterface
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.

A role can be associated to a SystemUnitClass by means of its CAEX SupportedRoleClass(es). This indicates that the class is able to support the referenced role(s). Multiple SupportedRoleClasses are supported. The MappingObject provides a mapping between role attributes and interfaces and SystemUnitClass attributes and interfaces. Once a SystemUnitClass is instantiated, the related InternalElement holds this information. However, this does not indicate that the instance actually plays all roles within its individual context.

CAEX RoleRequirements model the actual or requested roles of a CAEX InternalElement. Multiple RoleRequirements are supported. The actual role(s) are referenced, and requirements of the individual instance concerning the role are modelled within the RoleRequirement. The MappingObject allows mapping attributes and interfaces between the role class and the InternalElement, if required.

While associating a role class to an AML object, this AML object gets a semantic. Example role classes are a “Resource” or a “Robot”. Additional libraries are described in IEC 62714-2.

All described attributes are part of the AML standard library and may be removed after instantiation 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 9 to Figure 11 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 6.4.12.

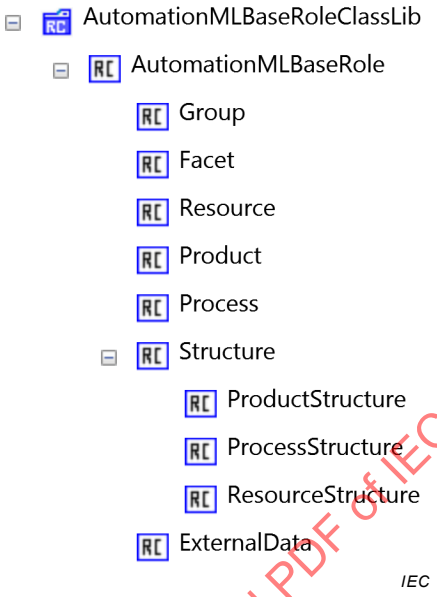


Figure 9 – AML basic role class library

RoleClassLib

Name	AutomationMLBaseRoleClassLib
Description	Automation Markup Language base role class library
Version	2.10.0

RoleClass

Name	AutomationMLBaseRole

RoleClass

Name	Group
RefBaseClassPath	AutomationMLBaseRole

Attribute

Name	AssociatedFacet
AttributeDataType	xs:string
RefAttributeType	AutomationMLBaseAttributeTypeLib/AssociatedFacet

RoleClass

Name	Facet
RefBaseClassPath	AutomationMLBaseRole

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	Structure
2	ProcessStructure	Structure
3	ResourceStructure	Structure

RoleClass

Name	ExternalData
RefBaseClassPath	AutomationMLBaseRole

IEC

Figure 10 – AutomationMLBaseRoleClassLib

```
<RoleClassLib Name="AutomationMLBaseRoleClassLib">
  <Description>Automation Markup Language base role class library</Description>
  <Version>2.10.0</Version>
  <RoleClass Name="AutomationMLBaseRole">
    <RoleClass Name="Group" RefBaseClassPath="AutomationMLBaseRole">
      <Attribute Name="AssociatedFacet" AttributeDataType="xs:string" RefAttributeType="AutomationMLBaseAttributeTypeLib/AssociatedFacet"/>
    </RoleClass>
    <RoleClass Name="Facet" RefBaseClassPath="AutomationMLBaseRole"/>
    <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="ExternalData" RefBaseClassPath="AutomationMLBaseRole"/>
  </RoleClass>
</RoleClassLib>
```

IEC

Figure 11 – 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
Path for element reference	AutomationMLBaseRoleClassLib/AutomationMLBaseRole
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
Path for element reference	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Group
Attributes	Name: AssociatedFacet RefAttributeType: AutomationMLBaseAttributeTypeLib/AssociatedFacet Semantics: see 6.5.2

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
Path for element reference	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Facet
Attributes	None

6.4.5 RoleClass Resource

Table 16 specifies the role class “Resource”.

Table 16 – 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.2.5.
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole
Path for element reference	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource
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).

6.4.6 RoleClass Product

Table 17 specifies the role class “Product”.

Table 17 – 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.2.5.
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole
Path for element reference	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Product
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).

6.4.7 RoleClass Process

Table 18 specifies the role class “Process”.

Table 18 – 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.2.5.
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole
Path for element reference	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Process
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).

6.4.8 RoleClass Structure

Table 19 specifies the role class “Structure”.

Table 19 – 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
Path for element reference	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure
Attributes	None

6.4.9 RoleClass ProductStructure

Table 20 specifies the role class “ProductStructure”.

Table 20 – 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
Path for element reference	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure/ ProductStructure
Attributes	None

6.4.10 RoleClass ProcessStructure

Table 21 specifies the role class “ProcessStructure”.

Table 21 – 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
Path for element reference	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure/ ProcessStructure
Attributes	None

6.4.11 RoleClass ResourceStructure

Table 22 specifies the role class “ResourceStructure”.

Table 22 – 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
Path for element reference	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure/ ResourceStructure
Attributes	None

6.4.12 RoleClass ExternalData

Table 23 specifies the role class “ExternalData”.

Table 23 – RoleClass ExternalData

Class name	ExternalData
Description	The role class “ExternalData” is an abstract role type for a document type and the base class for all document type roles. It describes different document types. AML document objects shall directly or indirectly reference this role. Examples are specified in A.1.5.
Parent class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole
Path for element reference	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/ExternalData
Attributes	None

6.5 AML basic attribute type library

6.5.1 General

Subclause 6.5 defines AML basic attribute types, required by the standard AML classes defined in 6.3 and 6.4. Figure 12 and Figure 13 present the standard basic attribute type library as XML table and as XML text. Details of each attribute type are given in 6.5.2.

AttributeTypeLib

Name	AutomationMLBaseAttributeTypeLib
Description	Standard Automation Markup Language Attribute Type Library
Version	2.10.0

AttributeType

Name	Direction
AttributeDataType	xs:string

Constraint

Name	AllowedValues

NominalScaledType

RequiredValue (3)

	Text
1	In
2	Out
3	InOut

AttributeType

Name	Cardinality

Attribute (2)

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

AttributeType

Name	Category
AttributeDataType	xs:string

AttributeType

Name	refURI
AttributeDataType	xs:anyURI

AttributeType

Name	AssociatedFacet
AttributeDataType	xs:string

AttributeType

Name	ListType

AttributeType

Name	OrderedListType

AttributeType

Name	LocalizedAttribute
AttributeDataType	xs:string

AttributeType

Name	AssociatedExternalValue

Attribute (3)

	Name	RefAttributeType
1	refCAEXAttribute	
2	refURI	AutomationMLBaseAttributeTypeLib/refURI
3	Direction	AutomationMLBaseAttributeTypeLib/Direction

AttributeType

Name	MIMEType
AttributeDataType	xs:string

AttributeType

Name	DocLang
AttributeDataType	xs:string

IEC

Figure 12 – AML basic attribute type library

```
<AttributeTypeLib Name="AutomationMLBaseAttributeTypeLib">
  <Description>Standard Automation Markup Language Attribute Type Library</Description>
  <Version>2.10.0</Version>
  <AttributeType Name="Direction" AttributeDataType="xs:string">
    <Constraint Name="AllowedValues">
      <NominalScaledType>
        <RequiredValue>In</RequiredValue>
        <RequiredValue>Out</RequiredValue>
        <RequiredValue>InOut</RequiredValue>
      </NominalScaledType>
    </Constraint>
  </AttributeType>
  <AttributeType Name="Cardinality">
    <Attribute Name="MinOccur" AttributeDataType="xs:unsignedInt"/>
    <Attribute Name="MaxOccur" AttributeDataType="xs:unsignedInt"/>
  </AttributeType>
  <AttributeType Name="Category" AttributeDataType="xs:string"/>
  <AttributeType Name="refURI" AttributeDataType="xs:anyURI"/>
  <AttributeType Name="AssociatedFacet" AttributeDataType="xs:string"/>
  <AttributeType Name="ListType"/>
  <AttributeType Name="OrderedListType"/>
  <AttributeType Name="LocalizedAttribute" AttributeDataType="xs:string"/>
  <AttributeType Name="AssociatedExternalValue">
    <Attribute Name="refCAEXAttribute"/>
    <Attribute Name="refURI" RefAttributeType="AutomationMLBaseAttributeTypeLib/refURI"/>
    <Attribute Name="Direction" RefAttributeType="AutomationMLBaseAttributeTypeLib/Direction"/>
  </AttributeType>
  <AttributeType Name="MIMEType" AttributeDataType="xs:string"/>
  <AttributeType Name="DocLang" AttributeDataType="xs:string"/>
</AttributeTypeLib>
```

IEC

Figure 13 – XML text of the AutomationMLBaseAttributeTypeLib

6.5.2 Attributes of the AutomationMLBaseAttributeTypeLib

Table 24 specifies the attribute types of the AutomationMLBaseAttributeTypeLib.

Table 24 – Attribute Types of the AutomationMLBaseAttributeTypeLib

Attribute name	Semantics
Direction	This attribute shall be used to describe a direction of a CAEX interface, e.g. of a signal or of an interface. Allowed values: “In”, “Out” or “InOut”. CAEX Interfaces using this attribute follow the following provisions: • Interfaces with the direction “In” shall only be connected to interfaces with the direction “Out” or “InOut”. • Interfaces with the direction “Out” shall only be connected to interfaces with the direction “In” or “InOut”. This information can be used e.g. in order to prove the validity of a connection. Examples: • Direction = “Out” (e.g. a plug) • Direction = “In” (e.g. a socket) • Direction = “InOut” NOTE The validity of those connections is outside the scope of IEC 62714, but is a tool functionality. AttributeDataType: xs:string Path: AutomationMLBaseAttributeTypeLib/Direction

Attribute name	Semantics
Cardinality	This attribute belongs to a CAEX ExternalInterface and shall be used to describe the allowed maximum and minimum numbers of connections from/to this interface. The attribute Cardinality itself is a complex attribute and shall not have a value. The corresponding sub-attributes are described in Table 25. AttributeDataType: This attribute has no AttributeDataType since the attribute has no value. Path: AutomationMLBaseAttributeTypeLib/Cardinality
Category	This attribute belongs to a CAEX ExternalInterface and describes the category of this interface. The value of this attribute is user-defined. Only interface classes with the same category value are allowed to be connected. This standard does not predefine category values. Example: Category = “MaterialFlow”. AttributeDataType: xs:string Path: AutomationMLBaseAttributeTypeLib/Category
refURI	This attribute shall be used in order to store a path to an external document. AttributeDataType: xs:anyURI Path: AutomationMLBaseAttributeTypeLib/refURI
AssociatedFacet	The attribute “AssociatedFacet” shall be used for the definition of the name of a related Facet. The Facet concept is described in 8.3. Example: AssociatedFacet = “PLCFacet”. AttributeDataType: xs:string Path: AutomationMLBaseAttributeTypeLib/AssociatedFacet
ListType	The attribute “ListType” shall be used for attributes that contain an unsorted list of attributes. The concept is described in A.2.7. AttributeDataType: empty Path: AutomationMLBaseAttributeTypeLib/ListType
OrderedListType	The attribute “OrderedListType” shall be used for attributes that contain a sorted list of attributes. The concept is described in A.2.7. AttributeDataType: empty Path: AutomationMLBaseAttributeTypeLib/OrderedListType
LocalizedAttribute	The attribute “LocalizedAttribute” shall be used for sub-attributes describing a language alternative of its parent attribute. The language according to RFC 5646 shall be used as name of the attribute. The concept is described in A.2.6. AttributeDataType: xs:string Path: AutomationMLBaseAttributeTypeLib/LocalizedAttribute
AssociatedValue	The AssociatedValue contains nested attributes which allow to interconnect a CAEX attribute to an item in an external document. The concept is described in A.1.6. The attribute AssociatedValue itself is a complex attribute and shall not have a value. The corresponding sub-attributes are described in Table 26. AttributeDataType: This attribute has no AttributeDataType since the attribute has no value. Parent: AutomationMLBaseAttributeTypeLib Path: AutomationMLBaseAttributeTypeLib/AssociatedValue

Attribute name	Semantics
MIMEType	The MIMEType describes the MIMEType of a referenced document. The attribute shall have values according to RFC 2046. Examples: MIMEType = “application/pdf” – this means that this document is of file type pdf. MIMEType = “application/xml” – this means that this document is of type xml. AttributeDataType: xs:string Parent: AutomationMLBaseAttributeTypeLib Path: AutomationMLBaseAttributeTypeLib/MIMEType
DocLang	The DocLang describes the language of a referenced document. The attribute shall have a value according to RFC 5646. Example: DocLang = “fr-FR”, “en-US”, “de-DE” – This means that this document is in French, American English or German language valid in France, US or Germany. AttributeDataType: xs:string Parent: AutomationMLBaseAttributeTypeLib Path: AutomationMLBaseAttributeTypeLib/DocLang

Table 25 – Sub-attributes of the attribute “Cardinality”

Attribute	AttributeDataType	Description	Example
MinOccur	xs:unsignedInt	The MinOccur value describes the minimum possible number of connections to or from the corresponding interface class. 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 the corresponding interface class. The attribute shall have values greater than or equal to Min-Occur, or 0 which means infinite.	MaxOccur = 3 This means that this Port can only be connected with a maximum of three other ports.

Table 26 – Sub-attributes of the attribute “AssociatedValue”

Attribute	RefAttributeType	Description	Example
refCAEXAttribute	<GUID/attName>	This attribute mirrors the CAEX attribute that should be interlinked with another item in an external document. This attribute has no value.	refCAEXAttribute= GUID1/Temperatur
refURI	AutomationMLBase-AttributeTypeLib/refURI	See refURI in Table 24	
Direction	AutomationMLBase-AttributeTypeLib/Direction	See Direction in Table 24	

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 SystemUnitClasses, CAEX AttributeTypes, CAEX InterfaceClasses and CAEX RoleClasses which are not predefined by IEC 62714 (all parts). 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 (all parts). 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 Attribute Library Concept or standard CAEX mappings. These concepts ease the automatic interpretation of user-defined classes, attribute types and attributes.

7.2 User-defined attributes

Attributes defined in IEC 62714 (all parts) are AML standard attributes. Attributes which are not defined in IEC 62714 (all parts) are user-defined attributes. All 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 62424:2016, 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 14 gives an example of a user-defined object “Object01” with a user-defined attribute “Length”.

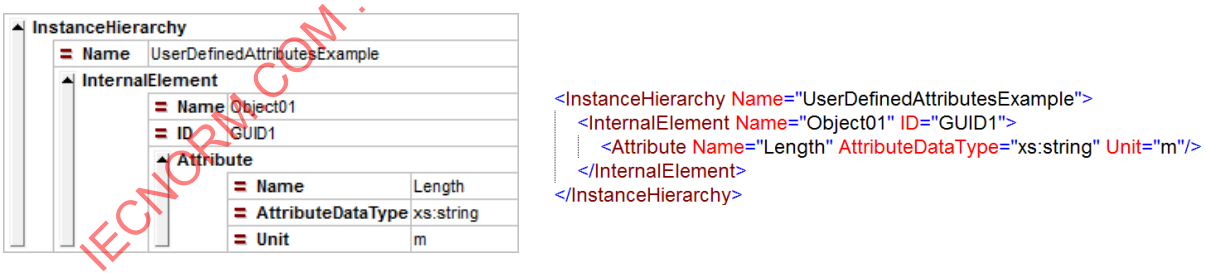


Figure 14 – Example of a user-defined attribute

7.3 User-defined AttributeTypes

Regarding user-defined AttributeTypes, the following provisions apply:

- User-defined AttributeTypes shall be stored in AML according to the CAEX AttributeType definition in IEC 62424:2016, A.2.5.
- AttributeTypeLibraries are recommended to model user specific, company specific, region specific, country specific, etc. or normative international standards with respect to attribute semantics, related to todays or future attribute standards. This is the basis for the storage of agreed attribute semantics and enables independence from language and naming conventions.

EXAMPLE: Figure 15 and Figure 16 illustrate the definition of user-defined AttributeTypes.

- The AttributeTypeLibrary “MyAttributeTypeLibrary” contains the AttributeTypes “Height”, “Width” and “Length”.
- The InternalElement “Station” has 3 attributes referencing the user-defined Attribute types.

NOTE The Attribute Type library concept allows to name object attributes different to the AttributeType. This enables independence from language and naming conventions.

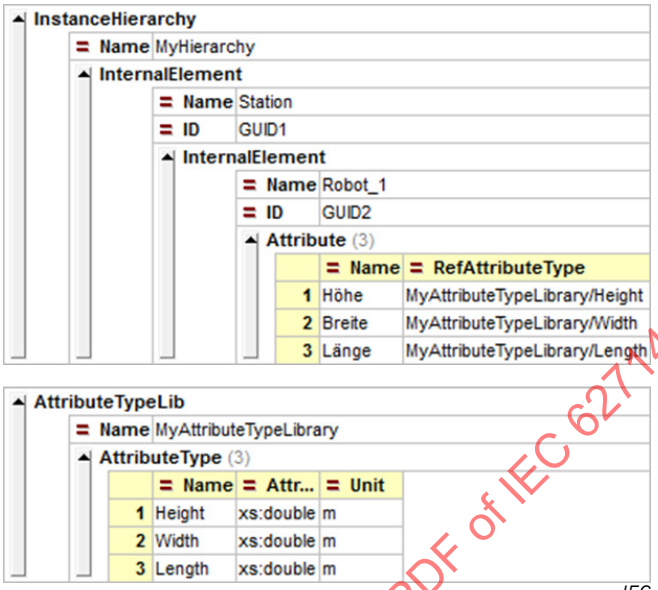


Figure 15 – Examples for user-defined AttributeTypes

```
<InstanceHierarchy Name="MyHierarchy">
  <InternalElement Name="Station" ID="GUID1">
    <InternalElement Name="Robot_1" ID="GUID2">
      <Attribute Name="Höhe" RefAttributeType="MyAttributeTypeLibrary/Height"/>
      <Attribute Name="Breite" RefAttributeType="MyAttributeTypeLibrary/Width"/>
      <Attribute Name="Länge" RefAttributeType="MyAttributeTypeLibrary/Length"/>
    </InternalElement>
  </InternalElement>
```

```
<AttributeTypeLib Name="MyAttributeTypeLibrary">
  <AttributeType Name="Height" AttributeDataType="xs:double" Unit="m"/>
  <AttributeType Name="Width" AttributeDataType="xs:double" Unit="m"/>
  <AttributeType Name="Length" AttributeDataType="xs:double" Unit="m"/>
</AttributeTypeLib>
```

IEC

Figure 16 – XML code of the examples for user-defined AttributeTypes

7.4 User-defined InterfaceClasses

All InterfaceClasses defined in IEC 62714 (all parts) are standard AML interface classes. All InterfaceClasses not defined in IEC 62714 (all parts) are 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 62424:2016, A.2.6.

  NOTE User-defined and standard AML interface classes 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 interface classes.

Figure 17 and Figure 18 show 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.

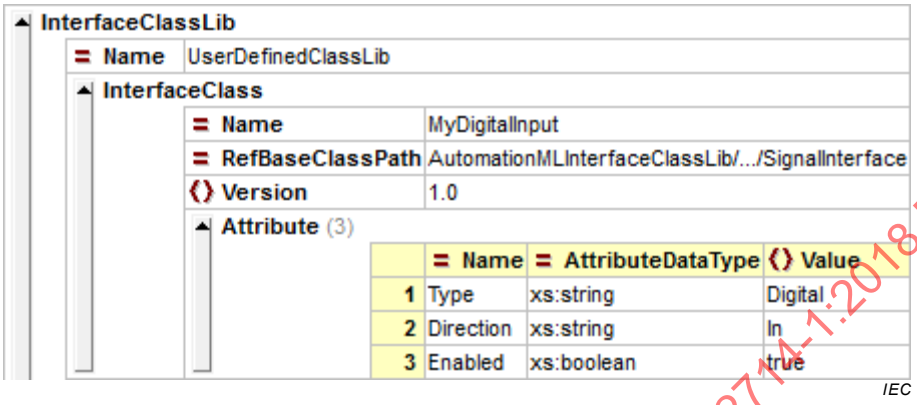


Figure 17 – Example of a user-defined InterfaceClass in a user-defined InterfaceClassLib

```
<InterfaceClassLib Name="UserDefinedClassLib">
  <InterfaceClass Name="MyDigitalInput" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface">
    <Version>1.0</Version>
    <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>
```

IEC

Figure 18 – XML code of the example of a user-defined InterfaceClass in a user-defined InterfaceClassLib

7.5 User-defined RoleClasses

All RoleClasses defined in IEC 62714 (all parts) are standard AML role classes. All role classes not defined in IEC 62714 (all parts) are user-defined RoleClasses.

Regarding user-defined RoleClasses, the following provisions apply:

- CAEX RoleClasses shall be stored according to the CAEX RoleClass definition in IEC 62424:2016, A.2.7.

  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 and Figure 20 show 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.

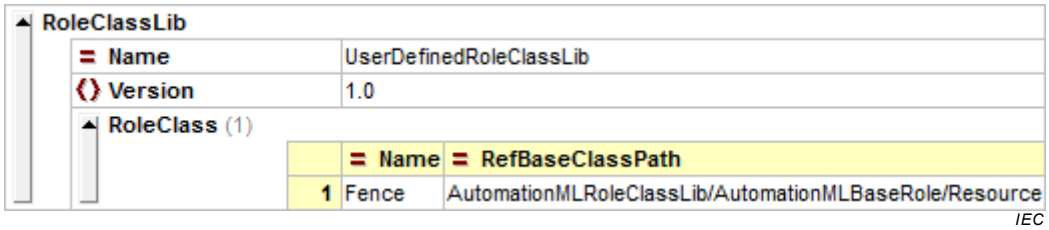


Figure 19 – Example of a user-defined RoleClass in a user-defined RoleClassLib

```
<RoleClassLib Name="UserDefinedRoleClassLib">
  <Version>1.0</Version>
  <RoleClass Name="Fence" RefBaseClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource"/>
</RoleClassLib>
```

IEC

Figure 20 – XML code of the example of a user-defined RoleClass in a user-defined RoleClassLib

7.6 User-defined SystemUnitClasses

IEC 62714 does not specify SystemUnitClasses, hence all SystemUnitClasses are userdefined.

Regarding user-defined SystemUnitClasses, the following provisions apply:

- User-defined SystemUnitClasses shall be stored in AML according to the CAEX SystemUnitClass definition in IEC 62424:2016, A.2.3.
- User-defined SystemUnitClasses should be assigned to a role class and shall use AML attributes whenever applicable. Hence, user-defined SystemUnitClasses may be assigned to a standard AML role class, a user-defined role class, or no role class.

It is recommended to assign user-defined SystemUnitClasses to a role class. This serves the automatic interpretability.

Examples: Figure 21 and Figure 22 illustrate 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 automatically be interpreted as "Resource".
- The SystemUnitClass "SpecialRobot1234" depicts a new user-defined class which is derived from "Robot1234". This class is therefore also a resource.

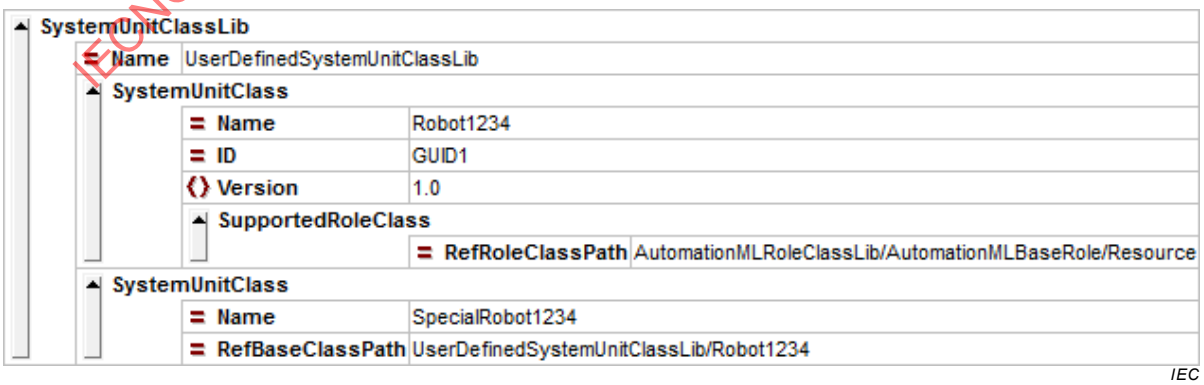


Figure 21 – Examples for different user-defined SystemUnitClasses

```
<SystemUnitClassLib Name="UserDefinedSystemUnitClassLib">
  <SystemUnitClass Name="Robot1234" ID="GUID1">
    <Version>1.0</Version>
    <SupportedRoleClass RefRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource"/>
  </SystemUnitClass>
  <SystemUnitClass Name="SpecialRobot1234" RefBaseClassPath="UserDefinedSystemUnitClassLib/Robot1234"/>
</SystemUnitClassLib>
```

IEC

Figure 22 – XML code of the examples for different user-defined SystemUnitClasses

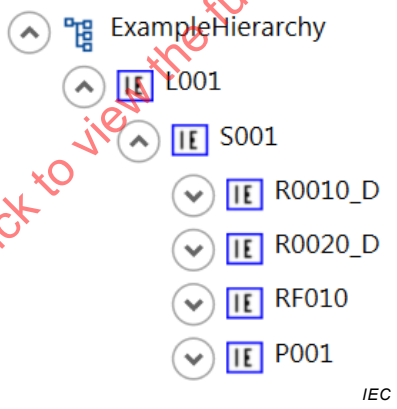
7.7 User-defined InstanceHierarchies

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 23 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”.



IEC

Figure 23 – Example of a user-defined InstanceHierarchy

Figure 24 shows the AML representation of this structure. According to IEC 62424:2016, A.2.10, 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="ABBRobotLibrary/ABB_Robot_1234">
        <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource/Mechatronic/Robot"/>
      </InternalElement>
      <InternalElement Name="R0020_D" ID="GUID4" RefBaseSystemUnitPath="ABBRobotLibrary/ABB_Robot_1234">
        <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource/Mechatronic/Robot"/>
      </InternalElement>
      <InternalElement Name="RF010" ID="GUID5">
        <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource/Mechatronic/Transport/Rollerbed"/>
      </InternalElement>
      <InternalElement Name="P001" ID="GUID6">
        <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource/ControlEquipment/ControlHardware/PLC"/>
      </InternalElement>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource/Structure/Station"/>
    </InternalElement>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource/Structure/Line"/>
  </InternalElement>
</InstanceHierarchy>
```

IEC

Figure 24 – 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 Clause A.2.

8.2 AML Port interface

An AML Port is a CAEX interface allowing the specification of complex (nested) 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 ExternalInterface with a direct or indirect association to the InterfaceClass “Port” which is described in 6.3.4.
- All nested ExternalInterfaces of the Port interface object should directly or indirectly be derived from an AML interface class defined in 6.3.

8.3 AML Facet object

A Facet is an AML object providing a sub-view on attributes or interfaces of the parent CAEX InternalElement. 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 a direct or indirect association to the RoleClass “Facet” which is described in 6.4.4.
- An AML Facet object may be modelled at an arbitrary position of the InstanceHierarchy or a SystemUnitClass and shall be a child object of an InternalElement or SystemUnitClass.
- Facets are identified by their unique ID. Their names are display names only.
- An InternalElement or SystemUnitClass may have an arbitrary number of Facet objects.
- An AML Facet object shall only contain mirror attributes or interfaces.
- Facets may have an arbitrary number of facet attributes and facet interfaces.
- Each Facet attribute shall mirror an existing attribute of the parent object, according to IEC 62424:2016, A.2.8.7. Mirroring of sub-attributes and other subordinated attributes within the parent object is possible.

- Facet attributes which are not part of the parent object are not permitted.
- Each Facet interface shall mirror an existing interface of the parent object, according to IEC 62424:2016, A.2.8.7. Mirroring of nested interfaces within the parent object is possible.
- 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 a direct or indirect association to the RoleClass “Group” which is defined in 6.4.3.
- An AML Group object may be modelled at an arbitrary position of a InstanceHierarchy or a SystemUnitClass and shall be a child object of an InstanceHierarchy, InternalElement or SystemUnitClass.
- Groups are identified by their unique ID. Their names are display names only.
- An InternalElement or SystemUnitClass may have an arbitrary number of Group objects.
- An AML Group object shall only contain mirror InternalElements and/or further Group objects.

  NOTE 1 Thus, Group objects can be nested.
- AML Groups shall not be used to describe plant hierarchies.
- An AML Group object may store additional information as attributes or interfaces in order to describe group specific information.

  NOTE 2 Those additional attributes or interfaces are not identical to attributes 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.
- If used, the attribute “AssociatedFacet” shall have a value that provides a valid name of an existing Facet of each mirror object.

8.5 Splitting of AML top-level data into different documents

According to IEC 62424:2016, 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.

8.6 Internationalization, AML multilingual expression

The AML multilingual expression concept allows to store textual expressions in different languages as one attribute structure. An informative overview about the multilingual expression concept including examples is provided in A.2.6.

Regarding AML multilingual expressions, the following provisions apply:

- A multilingual text attribute shall be modelled as CAEX Attribute. The value of this attribute shall represent the default expression. The default expression may be used if no specific language is requested or the requested language is not modelled.

- Each language expression of the attribute shall be modelled as nested attribute. Each of these child attributes shall reference the AttributeType “LocalizedAttribute”.
- The name of each child-attribute shall be the language of the expression in compliance with RFC 5646, e.g. “en-US”, “de-DE” or “fr-FR” etc. The value of the language attributes shall be the text in the related language.

Additional normative provisions regarding multilingual expressions are provided in 6.5.2.

8.7 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 62424:2016, 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.

8.8 Modelling of structured attribute lists or arrays

In many applications, the storage of lists is needed, e.g. a list of supported frequencies. AML allows the modelling and storage of list attributes and arrays. For the modelling of lists, the following provisions apply:

- A list is a sequence of homogenous items, i.e. all items shall be of the same data type.
- A list is modelled as CAEX Attribute that acts as root node for the list.
- If the list is not ordered, this list attribute shall reference to the AttributeType “ListType”.
- If the list is ordered, this list attribute shall reference to the AttributeType “OrderedListType”.
- The AttributeDataType, Value, DefaultValue and Unit of the list attribute shall be empty.
- The list items shall be modelled as child CAEX attributes of the list attribute.
- All child attributes shall be of the same data type.
- In case of an ordered list, the name of the child attributes shall be represented by an integer number “1”, “2” etc. For better readability, leading zeros can be appended, e.g. “0001”. The integer numbers shall represent the order index of each list item.

NOTE The term integer does not imply a data type.

- In case of a non-ordered list, the names of the child attributes shall be unique across the siblings.
- The child attributes may be again list attributes. This allows modelling arrays. In this case, all child attributes shall be referring to the AttributeType “ListType” or “OrderedListType”.

Subclause A.2.7 provides more information about lists, arrays and explains the modelling by means of examples.

8.9 AML Container

In order to transport an AML project containing multiple AML and other documents, this document supports the Open Packaging Conventions (OPC) as container format. Following the OPC definitions, the following provision apply:

- AML container shall be stored as OPC archive which provides data compression according to ISO/IEC 29500-2.
- AML container shall be either self-contained or environment-connected. Self-contained AML container shall include all related documents which are only locally interlinked with

each other. Environment-connected AML container may contain links to URIs to public documents outside of the AML container.

- AML container documents shall have the file extension ".amlx”.
- Following to the OPC convention, this document defines the following relationship types.

Root AML File:

A root AML file is an AML file acting as entry point into the AML container.

Relationshiptype: http://schemas.automationml.org/container/relationship/RootDocument

Mime type: "model/vnd.automationml+xml"

Library AML File:

A library AML file is an AML library which contains one or multiple elements of type Role-ClassLibrary, InterfaceClassLibrary, SystemUnitClassLibrary and/or AttributeTypeLibrary. Similar to the root AML file, the library AML file is an entry point into the AML container. AML containers may contain library files only.

Relationshiptype: http://schemas.automationml.org/container/relationship/Library

Mime type: "model/vnd.automationml+xml“

COLLADA File:

Relationshiptype: http://schemas.automationml.org/container/relationship/Collada

Mime type: "model/vnd.collada+xml"

PLCopenXML File:

Relationshiptype: http://schemas.automationml.org/container/relationship/PLCOpenXML

Mime type: "model/vnd.plcopen+xml"

Any Content:

Relationshiptype: http://schemas.automationml.org/container/relationship/AnyContent

Mime type: "application/x-any" or user-defined according to RFC 2046.

CAEX Schema:

Relationshiptype: http://schemas.automationml.org/container/relationship/CAEXSchema

Mime type: “text/xml”

PLCopenXML Schema:

Relationshiptype:
http://schemas.automationml.org/container/relationship/PCLOpenXMLSchema

Mime type: “text/xml”

COLLADA Schema:

Relationshiptype: http://schemas.automationml.org/container/relationship/ColladaSchema

Mime type: “text/xml”

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 interconnect 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 modelling 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. CAEX utilizes AML to interconnect 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.

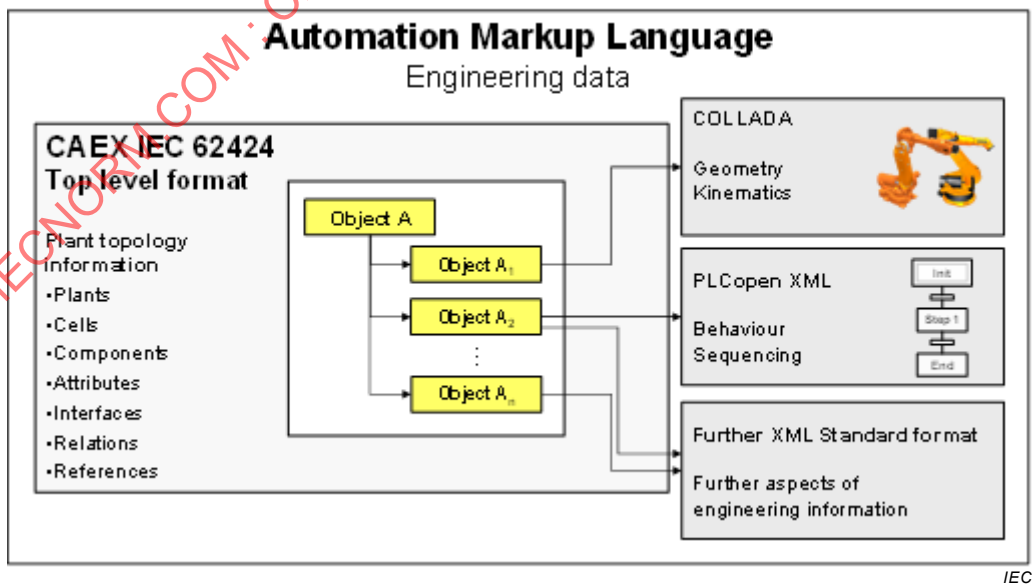


Figure A.1 – AML general architecture

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 62424:2016, 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 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 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 provided in IEC 62714-3 and IEC 62714-4. Subclause A.1.7 provides an informative overview about the modelling of references and relations with AML. Subclauses 5.5 and 5.6 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 3.1.19).

In order to store hierarchical plant structures, AML utilizes concepts provided by the top-level data format CAEX according to IEC 62424:2016, A.2.8.2. Figure A.2 shows an example of a plant topology of a manufacturing line containing several objects of different hierarchical levels.

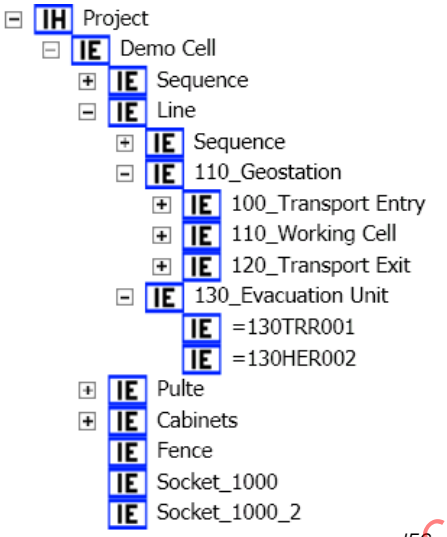


Figure A.2 – Plant topology with AML

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 data types for interfaces, roles, system units, attributes 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 elements and thus enable the automatic semantic interpretation of userdefined SystemUnitClasses or InternalElements. Subclause 6.4 specifies the basic AML-RoleClassLib with a set of abstract RoleClasses which are dedicated to general automation systems. Subclause 7.5 specifies the modelling of user-defined role classes. Further role libraries are defined in IEC 62714-2.
- **SystemUnits and SystemUnitClassLib**: The SystemUnitClassLib is used to model vendor specific components and their internal structure as classes. Subclause 7.6 specifies architecture rules for the definition of SystemUnitClasses. AML does not predefine a certain SystemUnitClassLib or SystemUnitClass.
- **AttributeTypes and AttributeTypeLib**: The AttributeTypeLib is used to model vendor specific Attribute Types or standardized Attribute Types. Attributes which are derived from an AttributeType inherit their semantics, even when they have a different name. This is the basis for language independence. Attribute type libraries can be utilized as basis for semantic libraries and serve for the automatic interpretation of vendor specific attributes.
- **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 7.7 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, IEC 62424:2016 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 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.

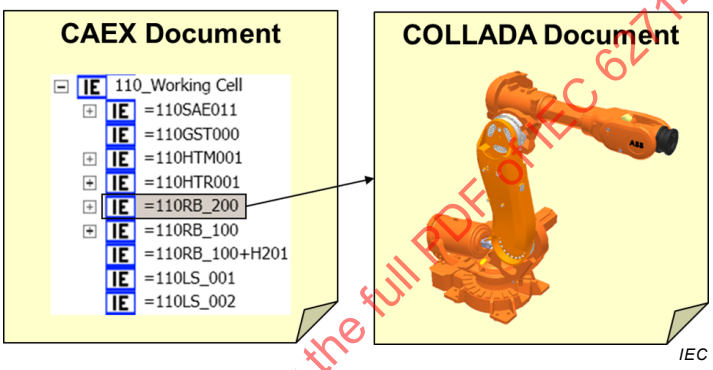


Figure A.3 – Reference from CAEX to a COLLADA document

The reference is modelled by means of a CAEX interface derived from the standard AML interface class "COLLADAInterface" specified in 5.6 and 6.3.7. Details are 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 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.

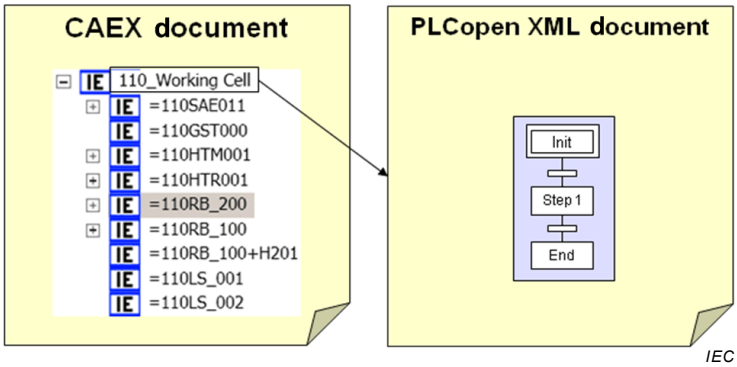


Figure A.4 – Reference from a CAEX to a PLCopen XML document

A.1.5 Referencing documents outside of the scope of IEC 62714

This document defines a mechanism which allows to reference external documents which are not in the scope of IEC 62714 (e.g. documents, pdf files, Excel sheets, XML files outside of the scope of IEC 62714, etc.). Figure A.5 and Figure A.6 illustrate how to model this.

In this example, the object “TemperatureSensor” should reference an external file “example.xml”. To model this reference, the InternalElement “TemperatureSensor” has a nested InternalElement “ParameterDocument” of type “ExternalData” representing the external document. Its attribute “DocLang” models the language of the document. The InternalElement “ParameterDocument” contains a CAEX ExternalInterface of type ExternalDataReference. Its attributes refURI and MIMEType define the URI of the document and the document type, in this case it is of the MIMEType “application/xml”. Normative provisions are defined in 5.6.5.

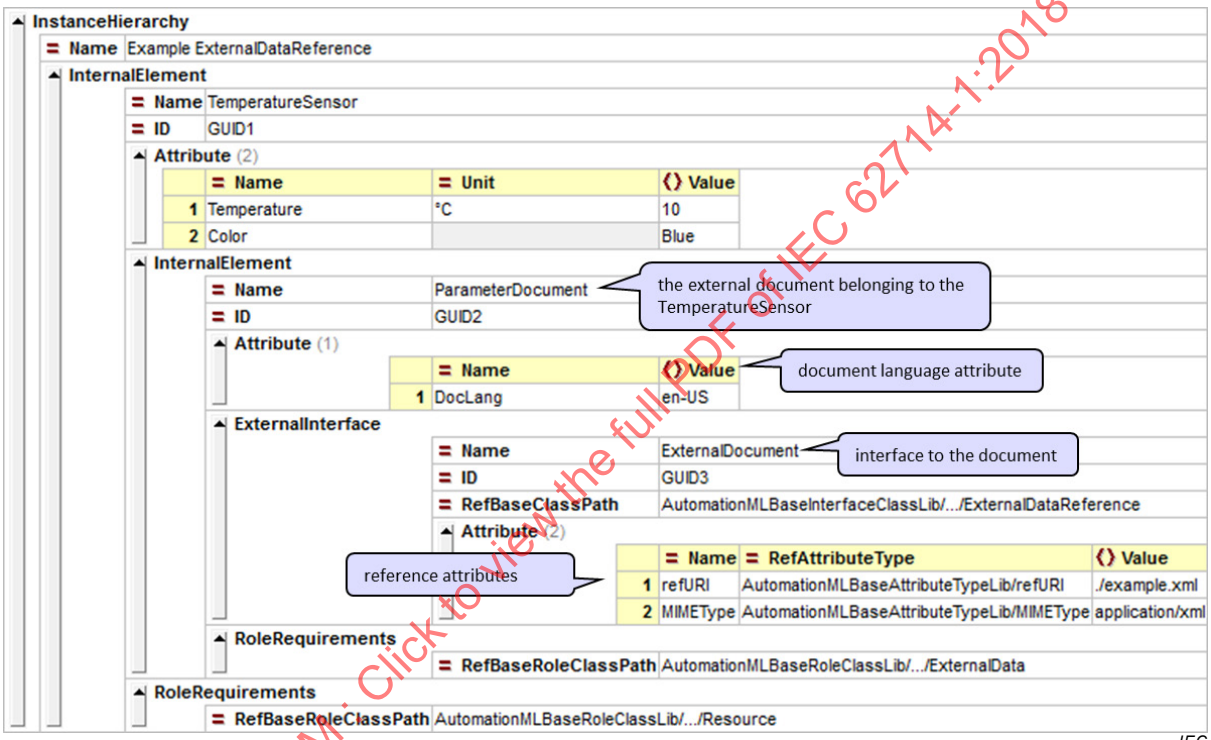


Figure A.5 – Example of referencing an external document

```
<InstanceHierarchy Name="Example ExternalDataReference">
  <InternalElement Name="TemperatureSensor" ID="GUID1">
    <Attribute Name="Temperature" Unit="°C">
      <Value>10</Value>
    </Attribute>
    <Attribute Name="Color">
      <Value>Blue</Value>
    </Attribute>
    <InternalElement Name="ParameterDocument" ID="GUID2">
      <Attribute Name="DocLang">
        <Value>en-US</Value>
      </Attribute>
      <ExternalInterface Name="ExternalDocument" ID="GUID3" RefBaseClassPath="AutomationMLBaseInterfaceClassLib/.../ExternalDataReference">
        <Attribute Name="refURI" RefAttributeType="AutomationMLBaseAttributeTypeLib/refURI">
          <Value>./example.xml</Value>
        </Attribute>
        <Attribute Name="MIMEType" RefAttributeType="AutomationMLBaseAttributeTypeLib/MIMEType">
          <Value>application/xml</Value>
        </Attribute>
      </ExternalInterface>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../ExternalData"/>
    </InternalElement>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
  </InternalElement>
</InstanceHierarchy>
```

IEC

Figure A.6 – XML text of the example for referencing an external document

A.1.6 Interlinking CAEX attributes and attributes in external documents

Figure A.7 and Figure A.8 illustrate how to model the interlinking between a CAEX attribute and an item in an external document. This example is an extension of the example shown in A.1.5: the Attribute “Temperature” should be referenced to an item in the external document “example.xml”.

In addition to the example shown in A.1.5/Figure A.5., the ExternalInterface “ExternalDocument” models a further attribute of type “AssociatedExternalValue”. This attribute contains 3 nested attributes.

- The attribute “refCAEXAttribute” mirrors the desired CAEX attribute. This attribute has no value.
- The attribute “refURI” references the external attribute. It references the same external document as the refURI attribute of the ExternalInterface: the “example.xml”.
- The attribute “Direction” provides the direction of the referencing, in this example the external document value is leading, CAEX is the consumer of the external value.

Normative provisions are defined in 5.6.6.

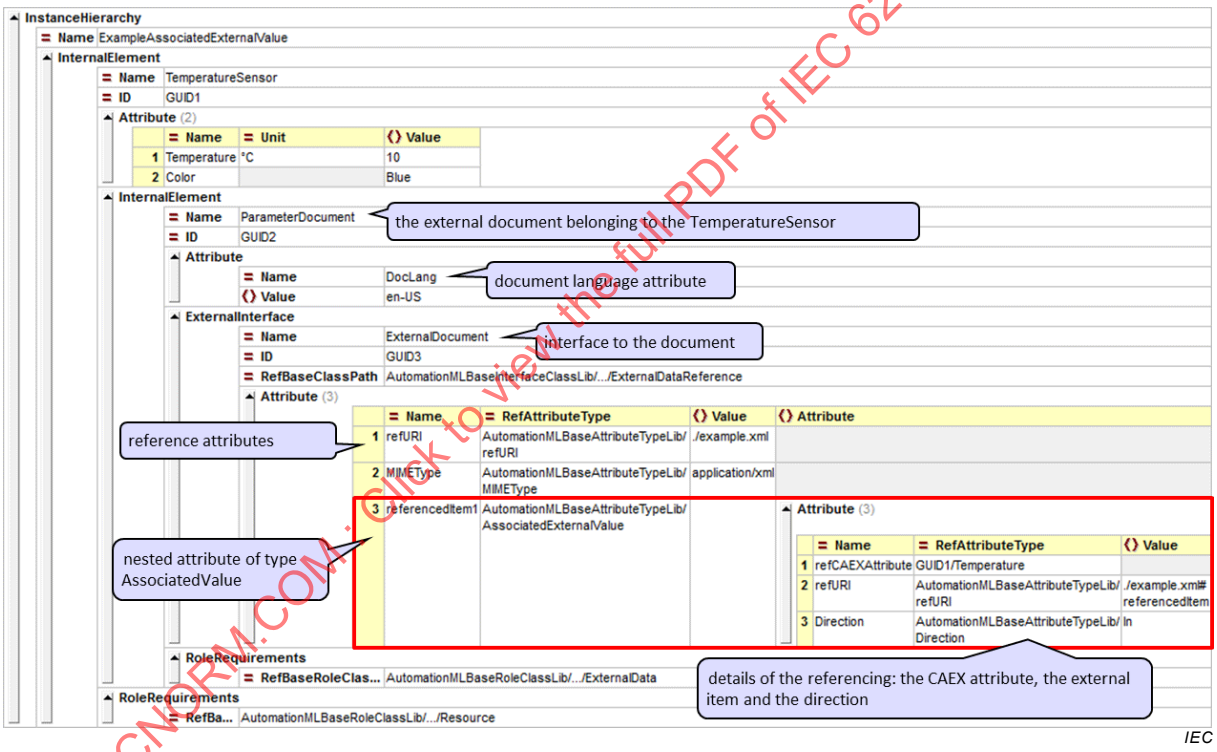


Figure A.7 – Example of referencing a CAEX attribute to an item in an external document

```
<InstanceHierarchy Name="ExampleAssociatedExternalValue">
  <InternalElement Name="TemperatureSensor" ID="GUID1">
    <Attribute Name="Temperature" Unit="°C">
      <Value>10</Value>
    </Attribute>
    <Attribute Name="Color">
      <Value>Blue</Value>
    </Attribute>
    <InternalElement Name="ParameterDocument" ID="GUID2">
      <Attribute Name="DocLang">
        <Value>en-US</Value>
      </Attribute>
      <ExternalInterface Name="ExternalDocument" ID="GUID3" RefBaseClassPath="AutomationMLBaseInterfaceClassLib/.../ExternalDataReference">
        <Attribute Name="refURI" RefAttributeType="AutomationMLBaseAttributeTypeLib/refURI">
          <Value>./example.xml</Value>
        </Attribute>
        <Attribute Name="MIMEType" RefAttributeType="AutomationMLBaseAttributeTypeLib/MIMEType">
          <Value>application/xml</Value>
        </Attribute>
        <Attribute Name="referencedItem1" RefAttributeType="AutomationMLBaseAttributeTypeLib/AssociatedExternalValue">
          <Value/>
          <Attribute Name="refCAEXAttribute" RefAttributeType="GUID1/Temperature"/>
          <Attribute Name="refURI" RefAttributeType="AutomationMLBaseAttributeTypeLib/refURI">
            <Value>./example.xml#referencedItem</Value>
          </Attribute>
          <Attribute Name="Direction" RefAttributeType="AutomationMLBaseAttributeTypeLib/Direction">
            <Value>In</Value>
          </Attribute>
        </Attribute>
      </ExternalInterface>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../ExternalData"/>
    </InternalElement>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
  </InternalElement>
</InstanceHierarchy>
```

IEC

Figure A.8 – XML text of the example for referencing a CAEX attribute to an item in an external document

A.1.7 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. CAEX supports the following relations according to IEC 62424:2016, A.2.8.

- parent-child-relations
  - parent-child-relations between AML objects
  - parent-child-relations between AML classes
- inheritance relations
  - inheritance relations between SystemUnitClasses
  - inheritance relations between RoleClasses
  - inheritance relations between InterfaceClasses
  - inheritance relations between AttributeTypes4
- class-instance-relations (see 5.5.2)
  - 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
  - relations between an AttributeType and an attribute
- instance-instance-relations (see 5.5.3)
  - relations between AML objects
  - relations between published externally stored data

Figure A.9 presents the mentioned relation types supported by AML by means of an example.

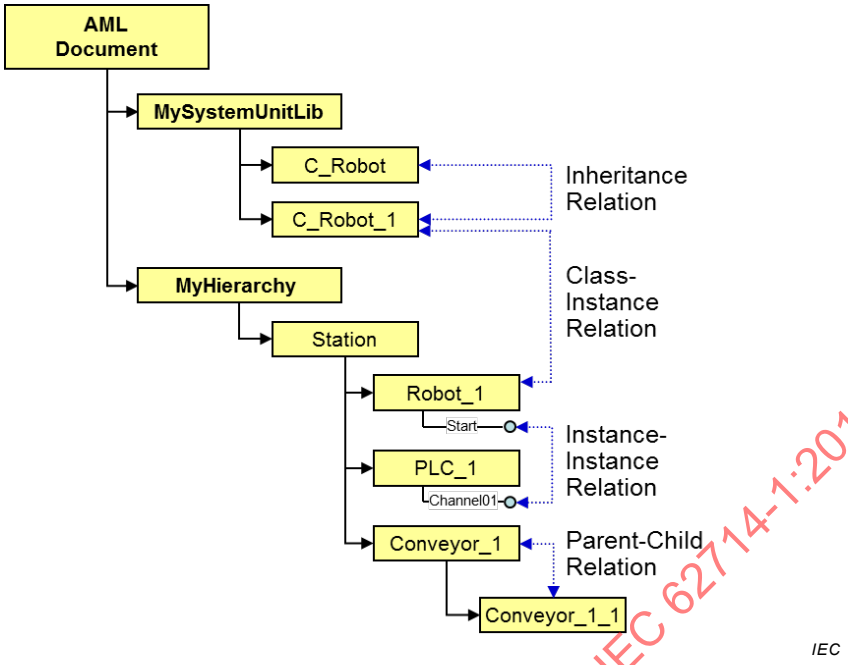


Figure A.9 – Relations in AML

Figure A.10 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.11 shows the corresponding XML text of the AML library.

Figure A.12 shows the XML text of the InstanceHierarchy.

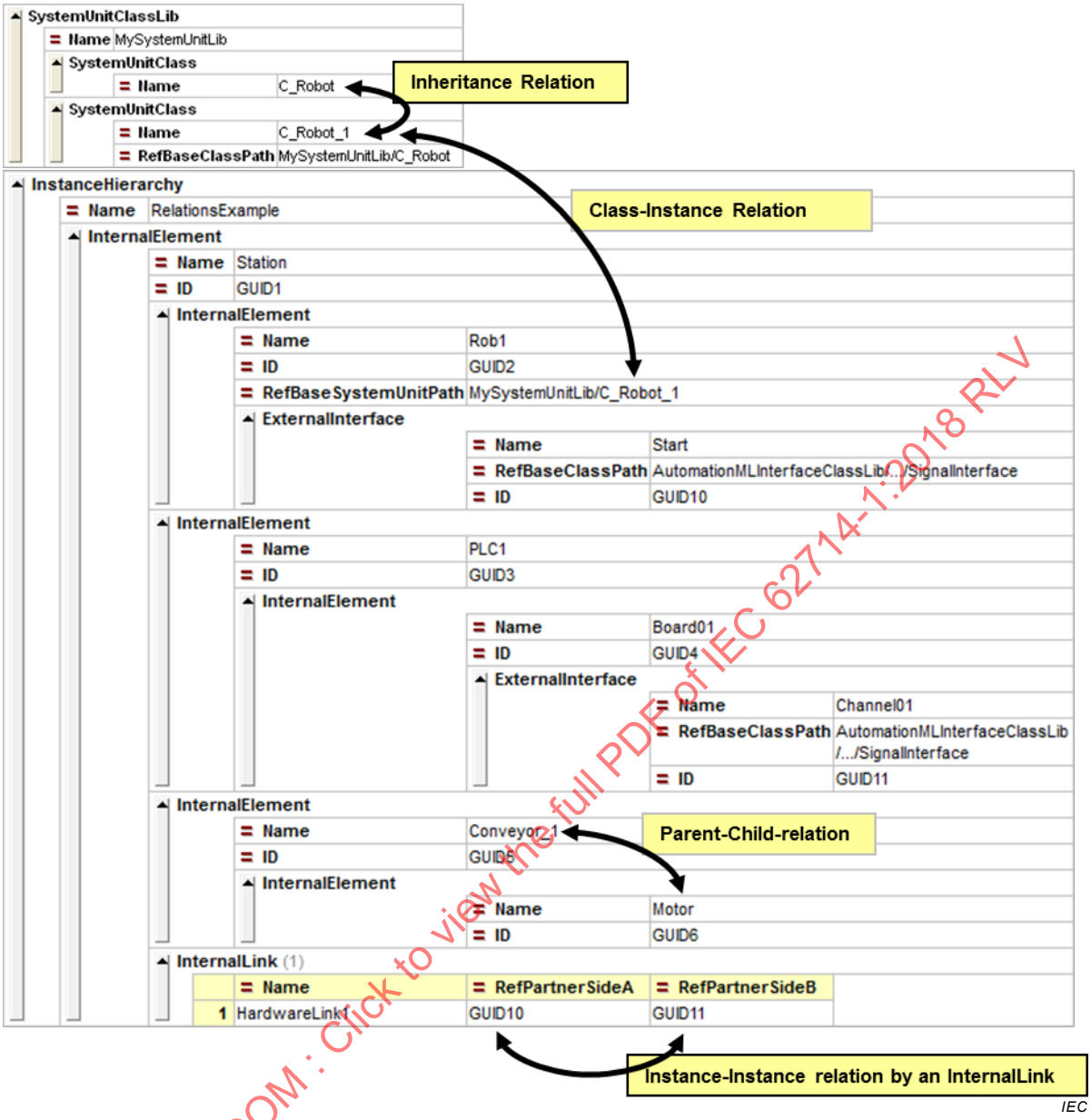


Figure A.10 – XML description of the relations example

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

IEC

Figure A.11 – 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" ID="GUID10"/>
    </InternalElement>
    <InternalElement Name="PLC1" ID="GUID3">
      <InternalElement Name="Board01" ID="GUID4">
        <ExternalInterface Name="Channel01" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface" ID="GUID11"/>
      </InternalElement>
    </InternalElement>
    <InternalElement Name="Conveyor_1" ID="GUID5">
      <InternalElement Name="Motor" ID="GUID6"/>
    </InternalElement>
    <InternalLink Name="HardwareLink1" RefPartnerSideA="GUID10" RefPartnerSideB="GUID11"/>
  </InternalElement>
</InstanceHierarchy>
```

IEC

Figure A.12 – 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.
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.
AML multilingual expression	AML multilingual expressions allow to store different languages for a textual expression.
List attribute and attribute arrays	AML allows modelling of attribute lists or arrays of attributes.

A.2.2 AML Port concept

A.2.2.1 Concept description

An AML Port is an AML interface that groups a number of nested interfaces (see Figure A.13). A Port object belongs to one parent 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 InterfaceClass “Port" (see 6.3.4). Normative provisions are specified in 8.2.

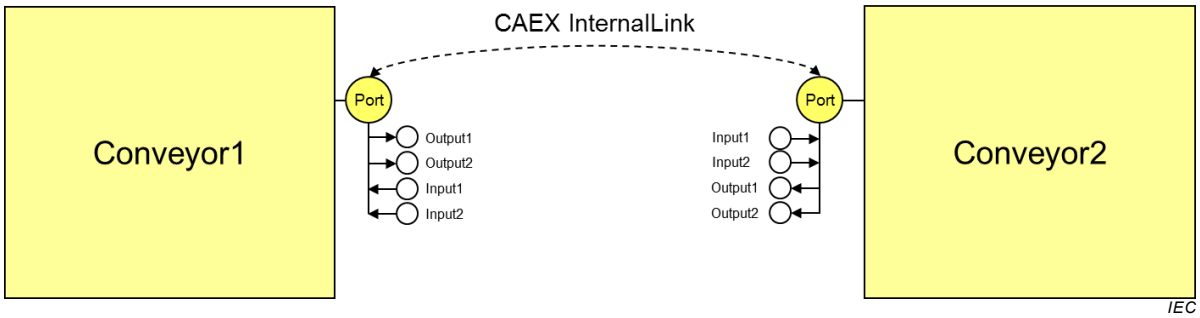


Figure A.13 – Port concept

A.2.2.2 Example

Figure A.14 gives an example for the AML Port concept. The object “Station” comprises the sub-objects “Conveyor1” and “Conveyor2”. Both sub-objects have a complex interface containing sub-interfaces. This is modelled by means of each a Port interface, derived from the AML standard InterfaceClass “Port”, and comprising a collection of nested interfaces. The 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 Ports are connected. In addition to this concept, AML allows modelling and storage of each individual link between the sub-interfaces.

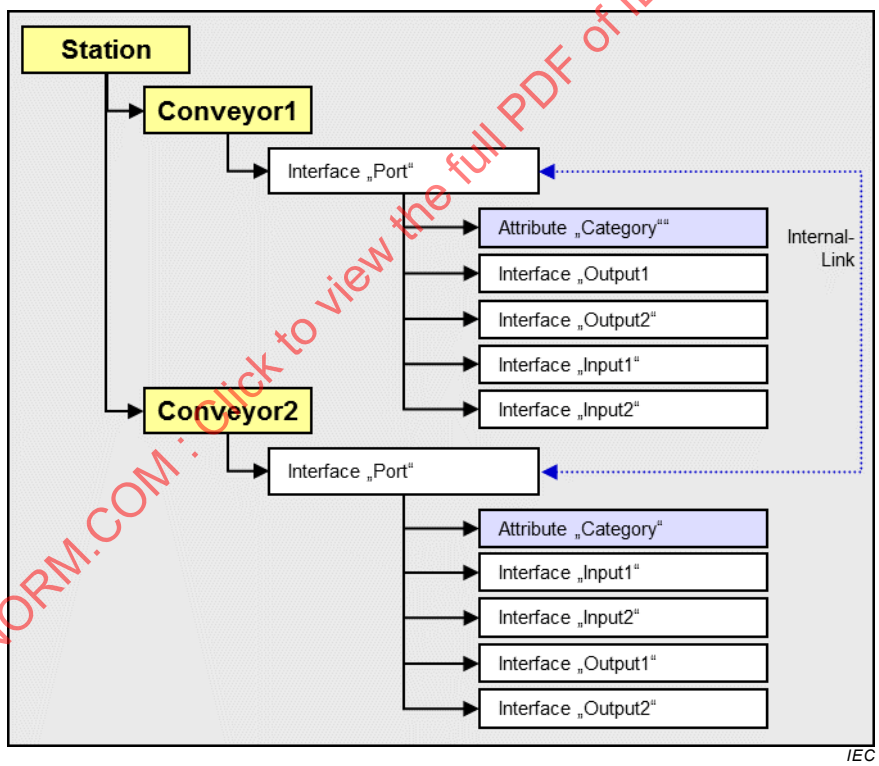


Figure A.14 – Example describing the AML Port concept

Figure A.15 and Figure A.16 describe the AML implementation of the example system described in Figure A.14.

- InstanceHierarchy
  - Name: PortExample AML 2.10
  - InternalElement
    - Name: Station
    - ID: GUID1
    - InternalElement
      - Name: Conveyor1
      - ID: GUID2
      - ExternalInterface
        - Name: Port
        - ID: GUID3
        - RefBaseClassPath: AutomationMLInterfaceClassLib/.../Port
        - Attribute
          - Name: Direction
          - AttributeDataType: xs:string
          - RefAttributeType: AutomationMLBaseAttributeTypeLib/Direction
          - Value: Out
        - Attribute
          - Name: Category
          - AttributeDataType: xs:string
          - RefAttributeType: AutomationMLBaseAttributeTypeLib/Category
          - Value: MaterialFlow
        - ExternalInterface (4)

	Name	RefBaseClassPath	ID
1	Output1	MyInterfaceClassLib/SignalInterface	GUID5
2	Output2	MyInterfaceClassLib/SignalInterface	GUID6
3	Input1	MyInterfaceClassLib/SignalInterface	GUID7
4	Input2	MyInterfaceClassLib/SignalInterface	GUID8

      - RoleRequirements
        - RefBaseRoleClassPath: AutomationMLRoleClassLib/AutomationMLBaseRole/Resource
    - InternalElement
      - Name: Conveyor2
      - ID: GUID4
      - ExternalInterface
        - Name: Port
        - ID: GUID10
        - RefBaseClassPath: AutomationMLInterfaceClassLib/.../Port
        - Attribute
          - Name: Direction
          - AttributeDataType: xs:string
          - RefAttributeType: AutomationMLBaseAttributeTypeLib/Direction
          - Value: In
        - Attribute
          - Name: Category
          - AttributeDataType: xs:string
          - RefAttributeType: AutomationMLBaseAttributeTypeLib/Category
          - Value: MaterialFlow
        - ExternalInterface (4)

	Name	RefBaseClassPath	ID
1	Input1	MyInterfaceClassLib/SignalInterface	GUID12
2	Input2	MyInterfaceClassLib/SignalInterface	GUID13
3	Output1	MyInterfaceClassLib/SignalInterface	GUID14
4	Output2	MyInterfaceClassLib/SignalInterface	GUID15

      - RoleRequirements
        - RefBaseRoleClassPath: AutomationMLRoleClassLib/AutomationMLBaseRole/Resource
    - InternalLink (1)

	Name	RefPartnerSideA	RefPartnerSideB
1	L1	GUID3	GUID10

IEC

Figure A.15 – XML description of the AML Port concept

```
<InstanceHierarchy Name="PortExample AML 2.10">
  <InternalElement Name="Station" ID="GUID1">
    <InternalElement Name="Conveyor1" ID="GUID2">
      <ExternalInterface Name="Port" ID="GUID3" RefBaseClassPath="AutomationMLInterfaceClassLib/.../Port">
        <Attribute Name="Direction" AttributeDataType="xs:string" RefAttributeType="AutomationMLBaseAttributeTypeLib/Direction">
          <Value>Out</Value>
        </Attribute>
        <Attribute Name="Category" AttributeDataType="xs:string" RefAttributeType="AutomationMLBaseAttributeTypeLib/Category">
          <Value>MaterialFlow</Value>
        </Attribute>
        <ExternalInterface Name="Output1" RefBaseClassPath="MyInterfaceClassLib/SignaleInterface" ID="GUID5"/>
        <ExternalInterface Name="Output2" RefBaseClassPath="MyInterfaceClassLib/SignaleInterface" ID="GUID6"/>
        <ExternalInterface Name="Input1" RefBaseClassPath="MyInterfaceClassLib/SignaleInterface" ID="GUID7"/>
        <ExternalInterface Name="Input2" RefBaseClassPath="MyInterfaceClassLib/SignaleInterface" ID="GUID8"/>
      </ExternalInterface>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource"/>
    </InternalElement>
    <InternalElement Name="Conveyor2" ID="GUID4">
      <ExternalInterface Name="Port" ID="GUID10" RefBaseClassPath="AutomationMLInterfaceClassLib/.../Port">
        <Attribute Name="Direction" AttributeDataType="xs:string" RefAttributeType="AutomationMLBaseAttributeTypeLib/Direction">
          <Value>In</Value>
        </Attribute>
        <Attribute Name="Category" AttributeDataType="xs:string" RefAttributeType="AutomationMLBaseAttributeTypeLib/Category">
          <Value>MaterialFlow</Value>
        </Attribute>
        <ExternalInterface Name="Input1" RefBaseClassPath="MyInterfaceClassLib/SignaleInterface" ID="GUID12"/>
        <ExternalInterface Name="Input2" RefBaseClassPath="MyInterfaceClassLib/SignaleInterface" ID="GUID13"/>
        <ExternalInterface Name="Output1" RefBaseClassPath="MyInterfaceClassLib/SignaleInterface" ID="GUID14"/>
        <ExternalInterface Name="Output2" RefBaseClassPath="MyInterfaceClassLib/SignaleInterface" ID="GUID15"/>
      </ExternalInterface>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Resource"/>
    </InternalElement>
    <InternalLink Name="L1" RefPartnerSideA="GUID3" RefPartnerSideB="GUID10"/>
  </InternalElement>
</InstanceHierarchy>
```

IEC

Figure A.16 – XML text describing the AML Port concept

A.2.2.3 Modelling a Port as user-defined Port

Ports can be extended by deriving an InterfaceClass. The following example in Figure A.17 depicts an XML description of a derived InterfaceClass “myPortInterface”. This class inherits all attributes of the standard class “Port” and adds a new attribute “Enabled”.

InterfaceClass		
= Name	UserDefinedPort	
= RefBaseClassPath	AutomationMLInterfaceClassLib/.../Port	
Attribute (1)		
	= Name	= AttributeDataType
1	Enabled	xs:boolean

```
<InterfaceClass Name="UserDefinedPort" RefBaseClassPath="AutomationMLInterfaceClassLib/.../Port">
  <Attribute Name="Enabled" AttributeDataType="xs:boolean"/>
</InterfaceClass>
```

IEC

Figure A.17 – Definition of a user-defined AML Port class “UserDefinedPort”

A.2.3 AML Facet concept

A.2.3.1 Concept description

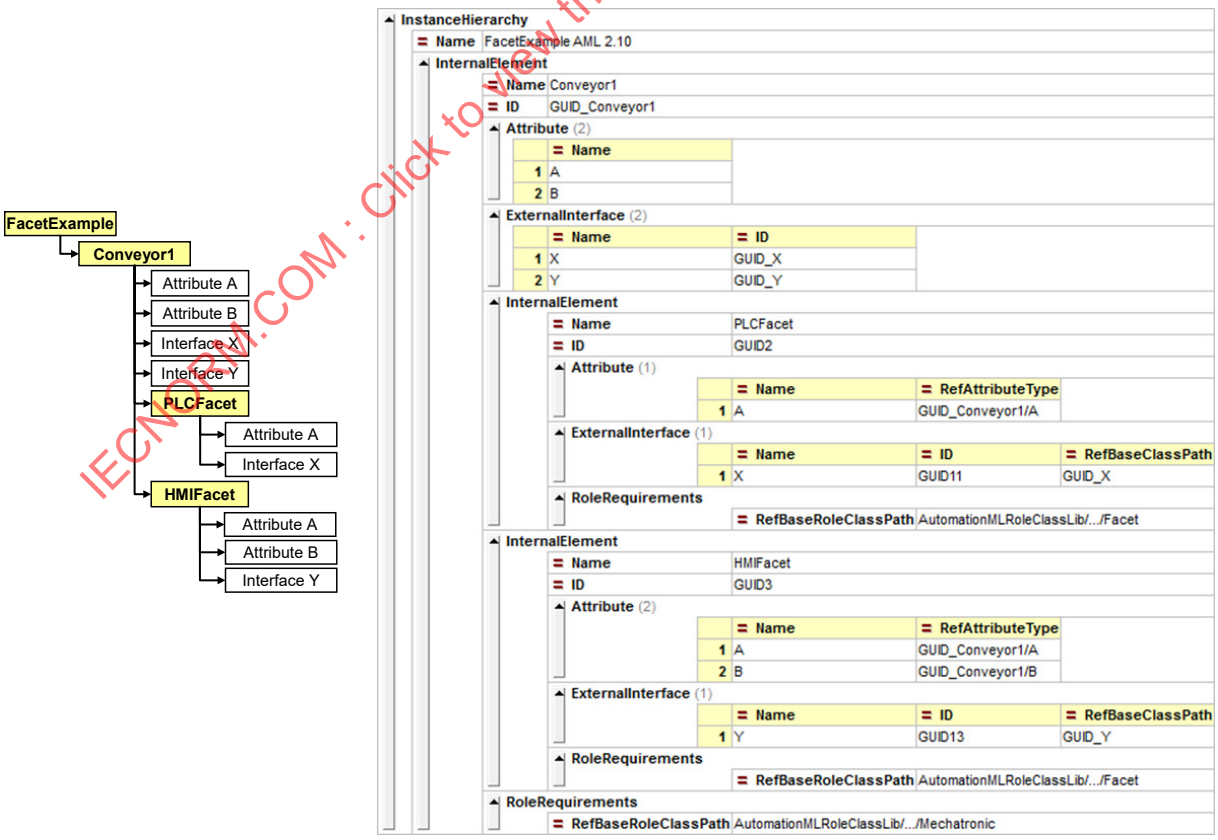
A Facet is an AML object providing a sub-view on attributes or interfaces of the parent AML object. This concept serves for the storage of different configuration settings such as HMI or PLC related data and allows the automation of several control engineering steps. For this, AML defines the AML RoleClass “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.18 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 described in A.2.4.4.



IEC

Figure A.18 – AML Facet example

```
<InstanceHierarchy Name="FacetExample AML 2.10">
  <InternalElement Name="Conveyor1" ID="GUID_Conveyor1">
    <Attribute Name="A"/>
    <Attribute Name="B"/>
    <ExternalInterface Name="X" ID="GUID_X"/>
    <ExternalInterface Name="Y" ID="GUID_Y"/>
    <InternalElement Name="PLCFacet" ID="GUID2">
      <Attribute Name="A" RefAttributeType="GUID_Conveyor1/A"/>
      <ExternalInterface Name="X" ID="GUID11" RefBaseClassPath="GUID_X"/>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../Facet"/>
    </InternalElement>
    <InternalElement Name="HMIFacet" ID="GUID3">
      <Attribute Name="A" RefAttributeType="GUID_Conveyor1/A"/>
      <Attribute Name="B" RefAttributeType="GUID_Conveyor1/B"/>
      <ExternalInterface Name="Y" ID="GUID13" RefBaseClassPath="GUID_Y"/>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../Facet"/>
    </InternalElement>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../Mechatronic"/>
  </InternalElement>
</InstanceHierarchy>
```

IEC

Figure A.19 – XML text of the AML Facet example

A.2.4 AML Group concept

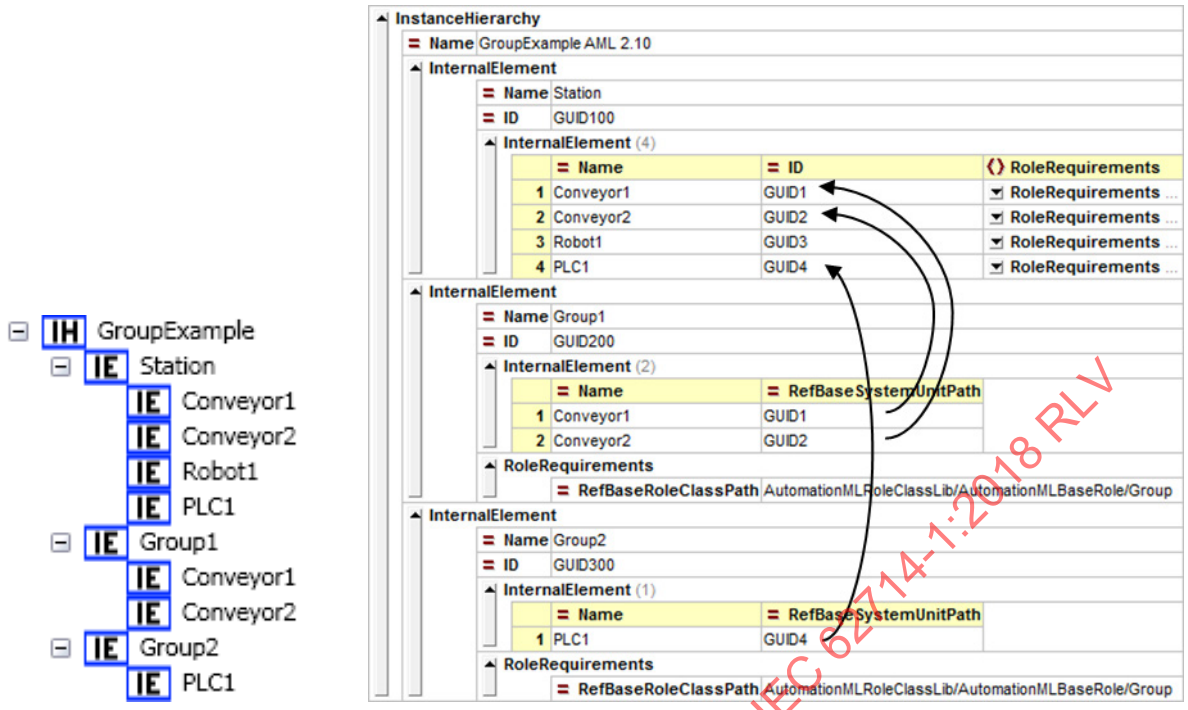
A.2.4.1 Concept description

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.20a) 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 62424:2016, A.2.8.7, CAEX provides the storage of such crossed structures. Figure A.20b) gives an AML implementation of this example and Figure A.21 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.20 – AML Group example

```
<InstanceHierarchy Name="GroupExample AML 2.10">
  <InternalElement Name="Station" ID="GUID100">
    <InternalElement Name="Conveyor1" ID="GUID1">
      <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../Transport/Rollerbed"/>
    </InternalElement>
    <InternalElement Name="Conveyor2" ID="GUID2">
      <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../Rollerbed"/>
    </InternalElement>
    <InternalElement Name="Robot1" ID="GUID3">
      <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../Robot"/>
    </InternalElement>
    <InternalElement Name="PLC1" ID="GUID4">
      <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../PLC"/>
    </InternalElement>
  </InternalElement>
  <InternalElement Name="Group1" ID="GUID200">
    <InternalElement Name="Conveyor1" RefBaseSystemUnitPath="GUID1"/>
    <InternalElement Name="Conveyor2" RefBaseSystemUnitPath="GUID2"/>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Group"/>
  </InternalElement>
  <InternalElement Name="Group2" ID="GUID300">
    <InternalElement Name="PLC1" RefBaseSystemUnitPath="GUID4"/>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/AutomationMLBaseRole/Group"/>
  </InternalElement>
</InstanceHierarchy>
```

IEC

Figure A.21 – XML text for the AML Group example

A.2.4.3 Combination of the Group and Facet concept

Figure A.22 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.

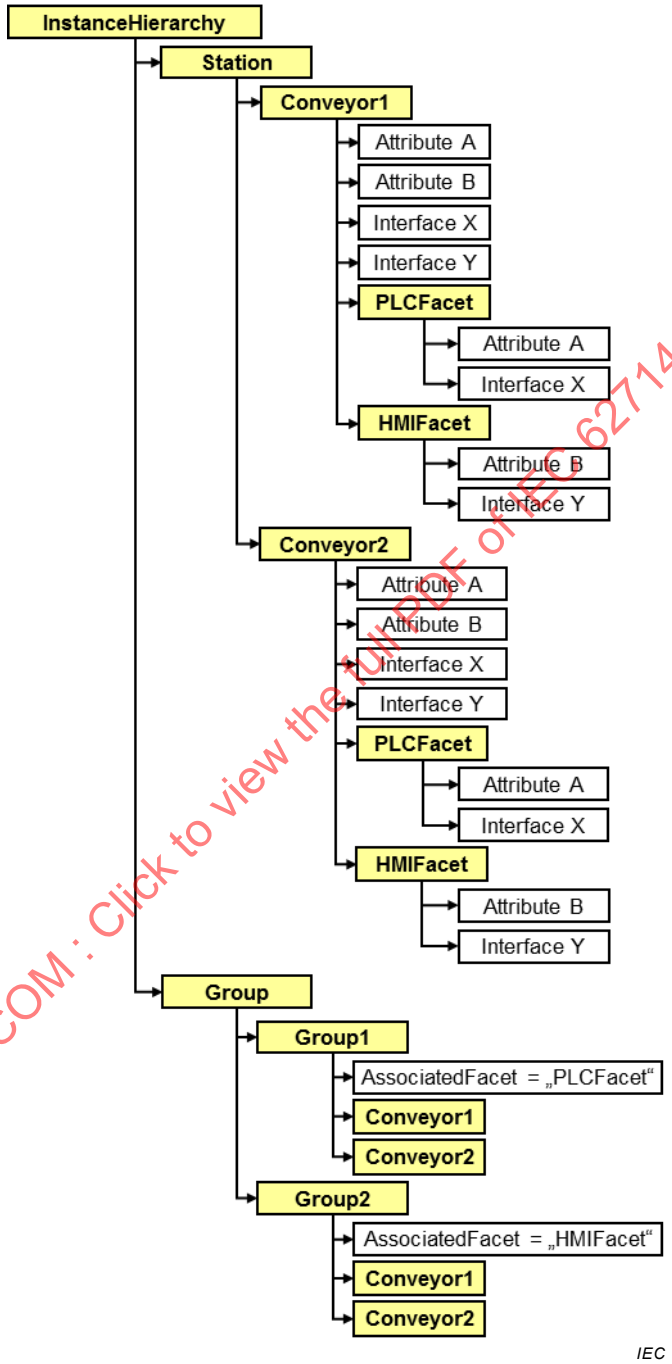


Figure A.22 – Combination of the Facet and Group concept

Figure A.23 shows the corresponding XML text related to the example shown in Figure A.22.

```
<InstanceHierarchy Name="FacetGroupCombination AML 2.10">
  <InternalElement Name="Station" ID="GUID0">
    <InternalElement Name="Conveyor1" ID="GUID1">
      <Attribute Name="A"/>
      <Attribute Name="B"/>
      <ExternalInterface Name="X" ID="GUID_X"/>
      <ExternalInterface Name="Y" ID="GUID_Y"/>
      <InternalElement Name="PLCFacet" ID="GUID2">
        <Attribute Name="A" RefAttributeType="GUID1/A"/>
        <ExternalInterface Name="X" ID="GUID11" RefBaseClassPath="GUID_X"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../Facet"/>
      </InternalElement>
      <InternalElement Name="HMIFacet" ID="GUID3">
        <Attribute Name="B" RefAttributeType="GUID1/B"/>
        <ExternalInterface Name="Y" ID="GUID12" RefBaseClassPath="GUID_Y"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../Facet"/>
      </InternalElement>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../Transport/Rollerbed"/>
    </InternalElement>
    <InternalElement Name="Conveyor2" ID="GUID4">
      <Attribute Name="A"/>
      <Attribute Name="B"/>
      <ExternalInterface Name="X" ID="GUID_X2"/>
      <ExternalInterface Name="Y" ID="GUID_Y2"/>
      <InternalElement Name="PLCFacet" ID="GUID5">
        <Attribute Name="A" RefAttributeType="GUID4/A"/>
        <ExternalInterface Name="X" ID="GUID13" RefBaseClassPath="GUID_X2"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../Facet"/>
      </InternalElement>
      <InternalElement Name="HMIFacet" ID="GUID6">
        <Attribute Name="B" RefAttributeType="GUID4/B"/>
        <ExternalInterface Name="Y" ID="GUID14" RefBaseClassPath="GUID_Y2"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../Facet"/>
      </InternalElement>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../Transport/Rollerbed"/>
    </InternalElement>
    <InternalElement Name="Group" ID="GUID7">
      <InternalElement Name="Group1" ID="GUID8">
        <Attribute Name="AssociatedFacet" RefAttributeType="AutomationMLBaseAttributeTypeLib/AssociatedFacet">
          <Value>HMIFacet</Value>
        </Attribute>
        <InternalElement Name="Conveyor1" RefBaseSystemUnitPath="GUID1"/>
        <InternalElement Name="Conveyor2" RefBaseSystemUnitPath="GUID4"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../Group"/>
      </InternalElement>
      <InternalElement Name="Group2" ID="GUID9">
        <Attribute Name="AssociatedFacet" RefAttributeType="AutomationMLBaseAttributeTypeLib/AssociatedFacet">
          <Value>PLCFacet</Value>
        </Attribute>
        <InternalElement Name="Conveyor1" RefBaseSystemUnitPath="GUID1"/>
        <InternalElement Name="Conveyor2" RefBaseSystemUnitPath="GUID4"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLRoleClassLib/.../Group"/>
      </InternalElement>
    </InternalElement>
  </InternalElement>
</InstanceHierarchy>
```

IEC

Figure A.23 – 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.24 illustrates this generic HMI template of a conveyor.

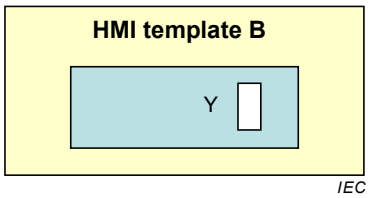


Figure A.24 – Generic HMI template "B" visualizing a process variable "Y" of a conveyor

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 extracts the HMI relevant information of both conveyors, identifies the corresponding HMI template and associates the correct signals to visualize.
Figure A.25 represents the resulting HMI.

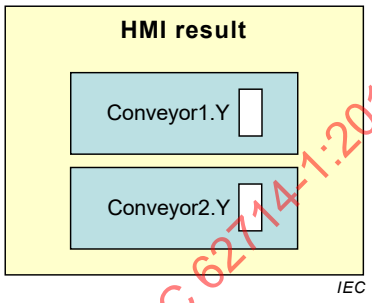


Figure A.25 – Generated HMI result "B" visualizing both conveyors with individual process variables

A.2.5 Process-Product-Resource concept

A.2.5.1 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.

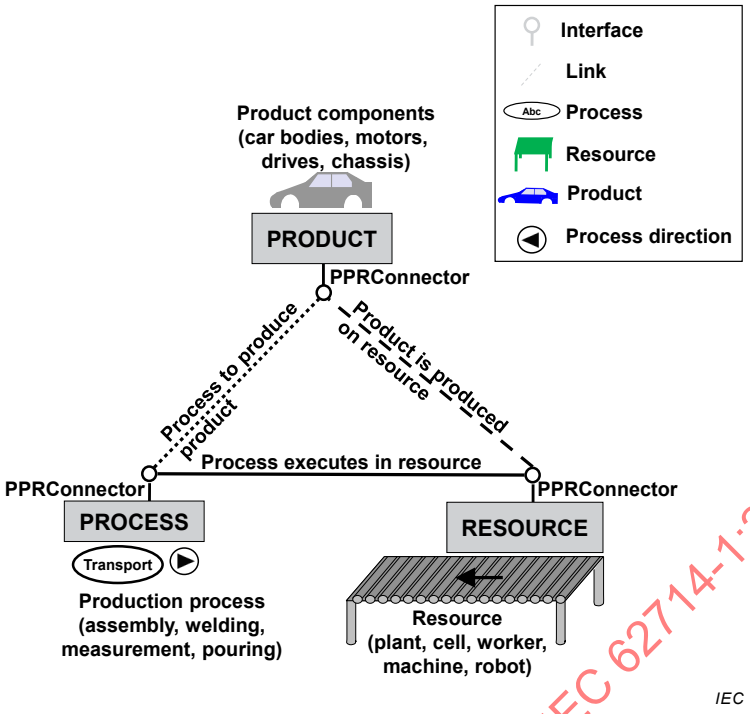


Figure A.26 – Base elements of the Product-Process-Resource concept

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 5.5.3). Thus, resources can be linked to products which they can manipulate.

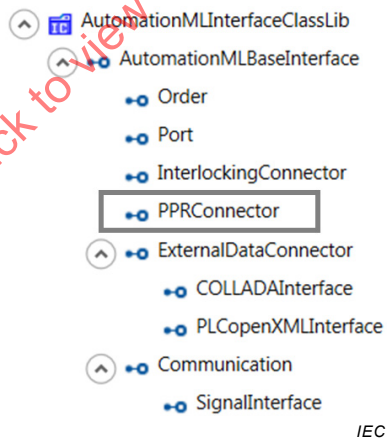


Figure A.27 – PPRConnector interface

A.2.5.2 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.

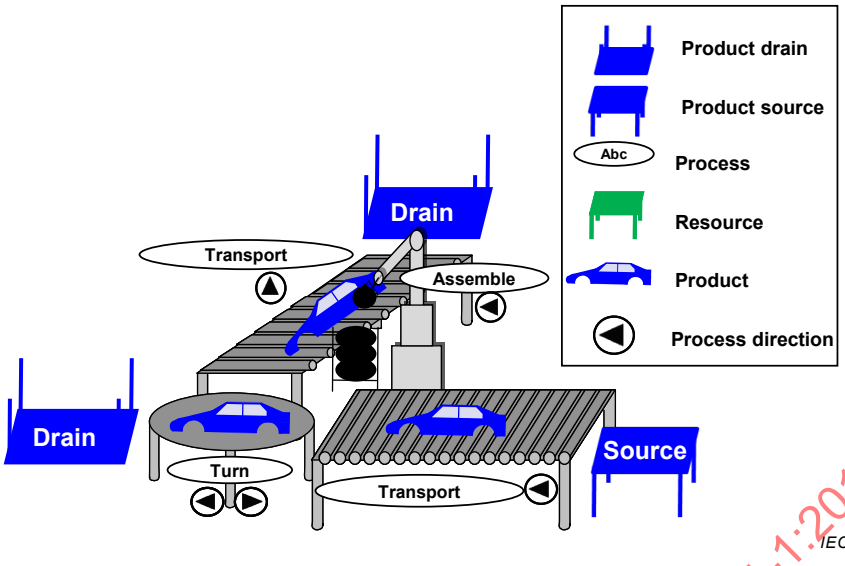


Figure A.28 – Example for the Product-Process-Resource concept

In AML, the Process-Product-Resource-concept is modelled by means of the CAEX role concept (see IEC 62424:2016, A.2.9) and relations between the elements (see 5.5.3). 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).

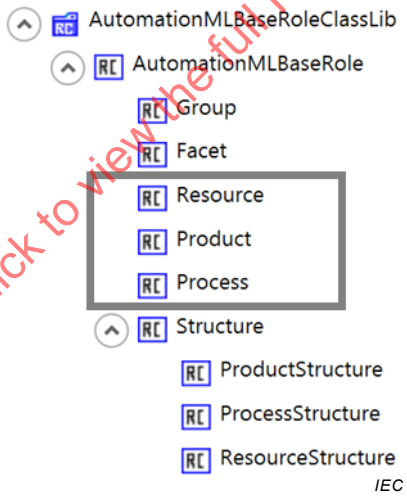


Figure A.29 – AML roles required for the Process-Product-Resource concept

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).

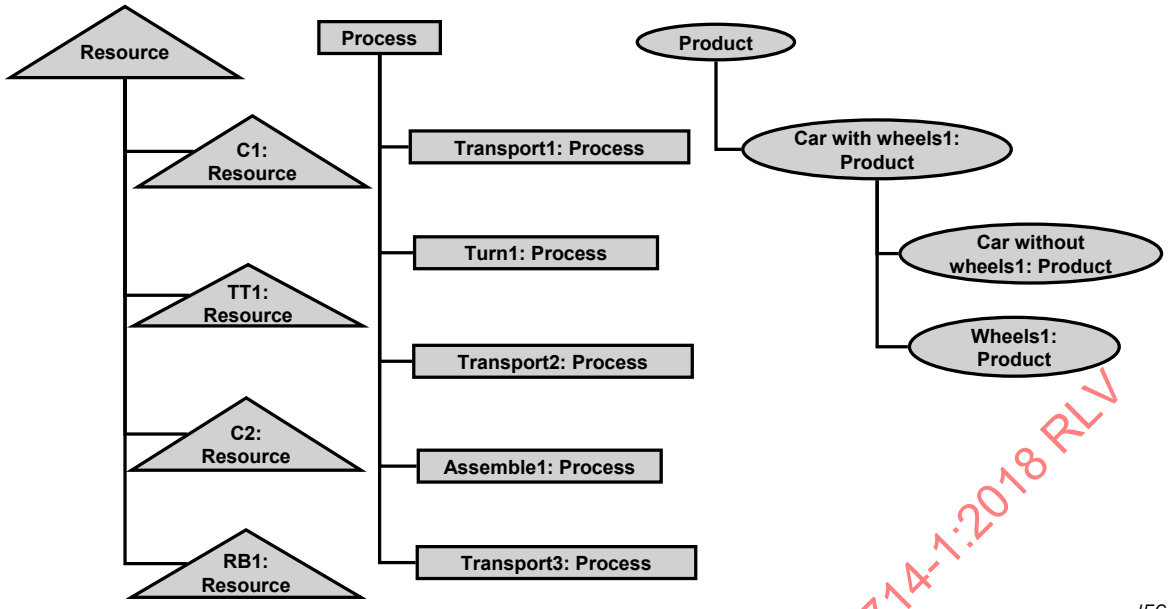


Figure A.30 – Elements of the example

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.

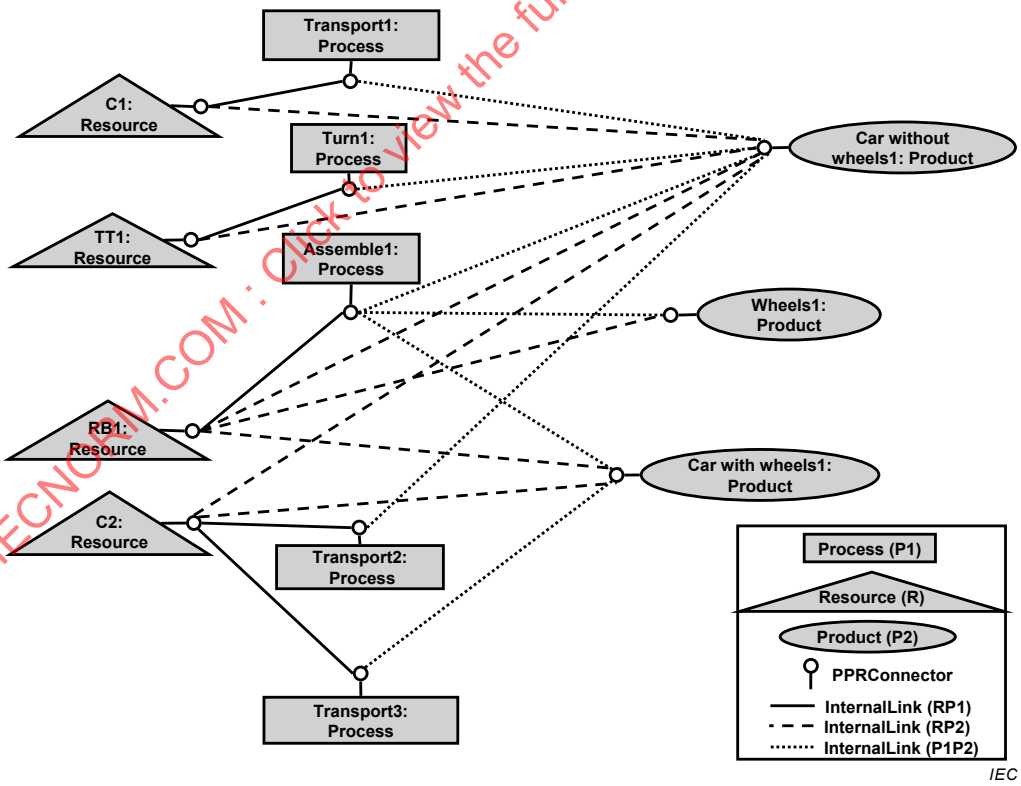


Figure A.31 – Links within the example

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.

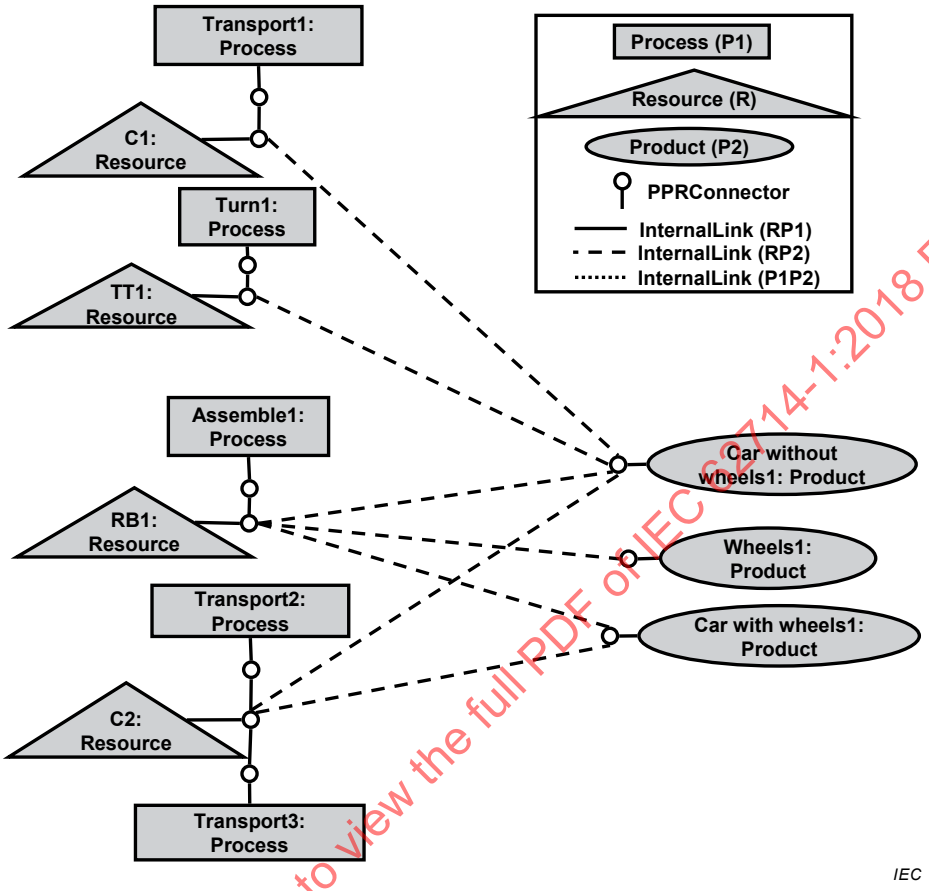


Figure A.32 – Links of the resource centric view on the example

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

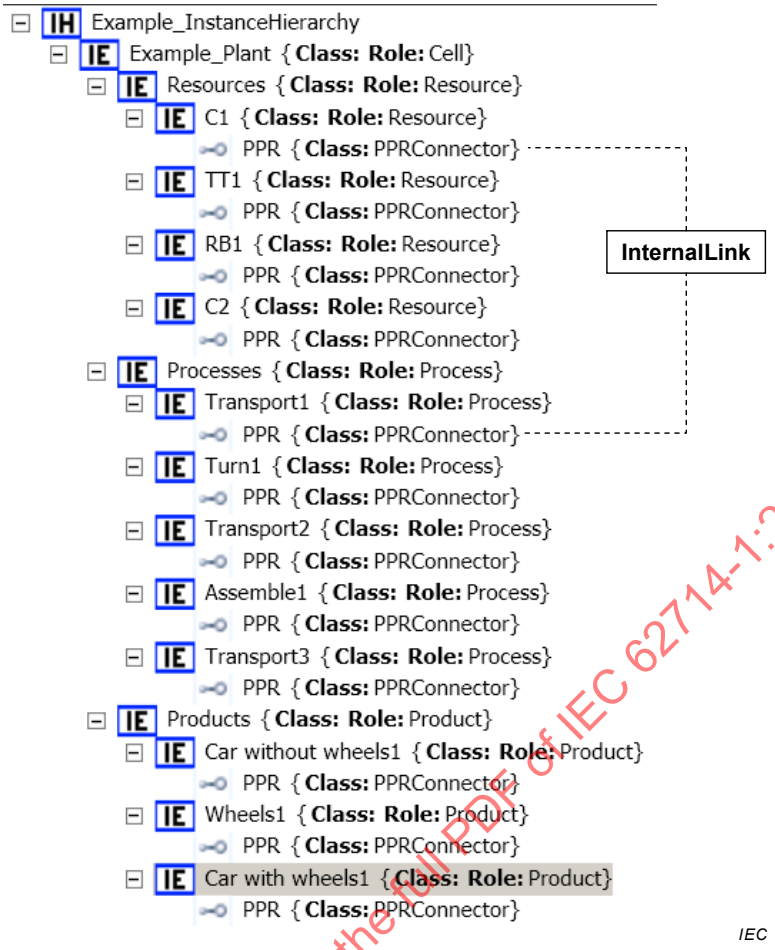


Figure A.33 – InstanceHierarchy of the example in AML

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.

InstanceHierarchy

= Name	Example_InstanceHierarchy

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	CarWithoutWheels1	GUID14
2	Wheels1	GUID15
3	CarWithWheels1	GUID16

IEC

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

IEC

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" ID="GUID3_PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
      </InternalElement>
      <InternalElement Name="TT1" ID="GUID4">
        <ExternalInterface Name="PPR" ID="GUID4_PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
      </InternalElement>
      <InternalElement Name="RB1" ID="GUID5">
        <ExternalInterface Name="PPR" ID="GUID5_PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
      </InternalElement>
      <InternalElement Name="C2" ID="GUID6">
        <ExternalInterface Name="PPR" ID="GUID6_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" ID="GUID8_PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Process"/>
      </InternalElement>
      <InternalElement Name="Turn1" ID="GUID9">
        <ExternalInterface Name="PPR" ID="GUID9_PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Process"/>
      </InternalElement>
      <InternalElement Name="Transport2" ID="GUID10">
        <ExternalInterface Name="PPR" ID="GUID10_PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Process"/>
      </InternalElement>
      <InternalElement Name="Assemble1" ID="GUID11">
        <ExternalInterface Name="PPR" ID="GUID11_PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Process"/>
      </InternalElement>
      <InternalElement Name="Transport3" ID="GUID12">
        <ExternalInterface Name="PPR" ID="GUID12_PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Process"/>
      </InternalElement>
      <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Process"/>
    </InternalElement>
    <InternalElement Name="Products" ID="GUID13">
      <InternalElement Name="CarWithoutWheels1" ID="GUID14">
        <ExternalInterface Name="PPR" ID="GUID14_PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Product"/>
      </InternalElement>
      <InternalElement Name="Wheels1" ID="GUID15">
        <ExternalInterface Name="PPR" ID="GUID15_PPR" RefBaseClassPath="AutomationMLInterfaceClassLib/.../PPRConnector"/>
        <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Product"/>
      </InternalElement>
      <InternalElement Name="CarWithWheels1" ID="GUID16">
        <ExternalInterface Name="PPR" ID="GUID16_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>
```

IEC

Figure A.36 – InstanceHierarchy of the example in XML

A.2.6 AML multilingual expression concept

A.2.6.1 Concept description

The concept of multilingual expressions aims for storing multi language information within the same AML file.

A.2.6.2 Example of a label attribute with 3 localizations

Figure A.37 gives an example for the AML multilingual expression. The object “PC123” comprises the attribute “Label". The attribute “Label” itself contains the expression in the default language. Modelling localized language texts requires nested attributes. For this purpose, the attribute “Label” comprises the sub-attributes "en-US", "de-DE", and "fr-FR". The names of the sub-attributes are corresponding to language codes according to RFC 5646, their values contain the corresponding localized texts.

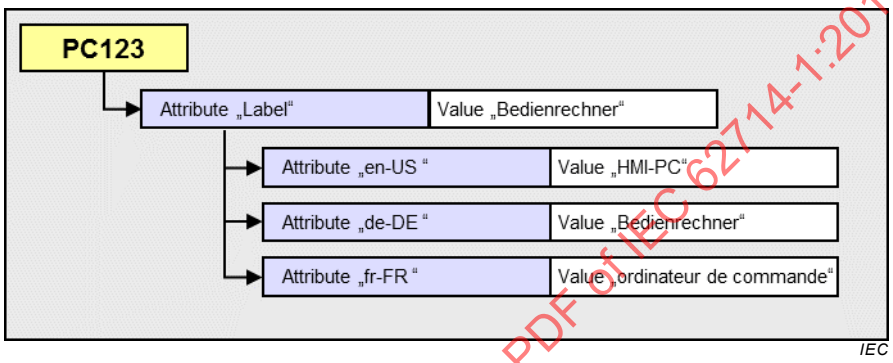


Figure A.37 – Example describing the AML multilingual expression concept

Figure A.38 shows the modelling of the example with AML according to the provisions described in 8.6.

InternalElement
- Name: PC123
- ID: GUID1
- Attribute
  - Name: Label
  - Description: Display name for this element. The sub attributes provide the language code according to RFC 5646.
  - Value: Bedienrechner
  - Attribute [3]

	Name	RefAttributeType	Value
1	en-US	AutomationMLBaseAttributeTypeLib/LocalizedAttribute	HMI-PC
2	de-DE	AutomationMLBaseAttributeTypeLib/LocalizedAttribute	Bedienrechner
3	fr-FR	AutomationMLBaseAttributeTypeLib/LocalizedAttribute	ordinateur de commande

- RoleRequirements
  - RefBaseRoleClassPath: AutomationMLBaseRoleClassLib/.../Resource

IEC

Figure A.38 – XML description of the AML multilingual expression concept

Figure A.39 shows the corresponding XML code of this example.

```
<InternalElement Name="PC123" ID="GUID1">
    <Attribute Name="Label">
        <Description>Display name for this element. The sub attributes provide the language code according to RFC 5646.</Description>
        <Value>Bedienrechner</Value>
        <Attribute Name="en-US" RefAttributeType="AutomationMLBaseAttributeTypeLib/LocalizedAttribute">
            <Value>HMI-PC</Value>
        </Attribute>
        <Attribute Name="de-DE" RefAttributeType="AutomationMLBaseAttributeTypeLib/LocalizedAttribute">
            <Value>Bedienrechner</Value>
        </Attribute>
        <Attribute Name="fr-FR" RefAttributeType="AutomationMLBaseAttributeTypeLib/LocalizedAttribute">
            <Value>ordinateur de commande</Value>
        </Attribute>
    </Attribute>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
</InternalElement>
```

IEC

Figure A.39 – XML text describing the AML multilingual expression concept

A.2.6.3 Example of a label attribute type with 3 localizations

Another use case is the predefinition of attributes including all required localized language expressions in a library. Thus, if the label of a specific object, e.g. an HMI-PC, is frequently used, it is useful to model it in a library. Figure A.40 illustrates this by the CAEX AttributeType *HMI-PC-LabelType*. It predefines all 3 localized language expressions.

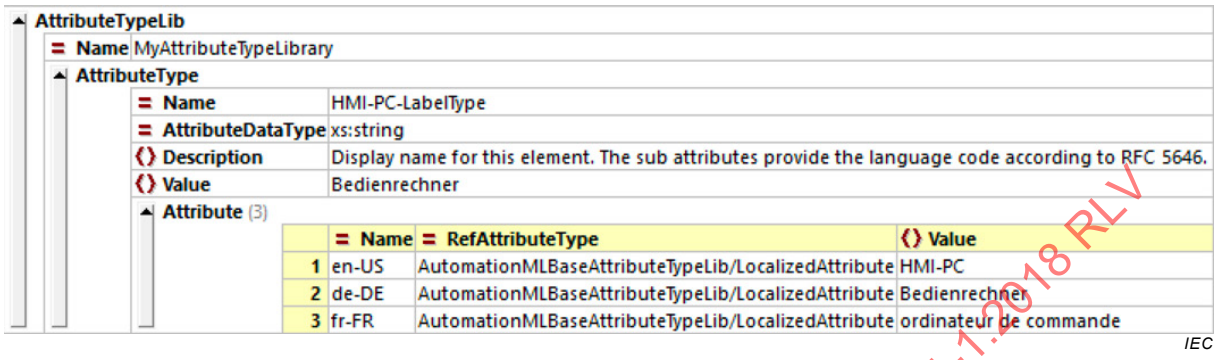


Figure A.40 – AML model of a multilingual AttributeType

Figure A.41 shows the related XML code of the given example.

```
<AttributeTypeLib Name="MyAttributeTypeLibrary">
    <AttributeType Name="HMI-PC-LabelType" AttributeDataType="xs:string">
        <Description>Display name for this element. The sub attributes provide the language code according to RFC 5646.</Description>
        <Value>Bedienrechner</Value>
        <Attribute Name="en-US" RefAttributeType="AutomationMLBaseAttributeTypeLib/LocalizedAttribute">
            <Value>HMI-PC</Value>
        </Attribute>
        <Attribute Name="de-DE" RefAttributeType="AutomationMLBaseAttributeTypeLib/LocalizedAttribute">
            <Value>Bedienrechner</Value>
        </Attribute>
        <Attribute Name="fr-FR" RefAttributeType="AutomationMLBaseAttributeTypeLib/LocalizedAttribute">
            <Value>ordinateur de commande</Value>
        </Attribute>
    </AttributeType>
</AttributeTypeLib>
```

IEC

Figure A.41 – XML code of the a multilingual AttributeType

Wherever the label of a HMI-PC is needed including localized language expressions, it needs to reference this attribute type and gets all predefined localized language expressions. This technique can be used to model arbitrary project specific multilingual expressions in a library.

A.2.7 Attribute lists and arrays

A.2.7.1 Concept description

In practice, attributes are often structured. Lists or arrays of data types have to be modelled and exchanged between engineering tools. A list consists of items of the same data type, and may contain further lists, enabling the modelling of arrays of arbitrary dimensions.

Finally, a list is modelled in AML as a *list attribute* containing nested child attributes. The list attribute acts as container for the list. In order to declare a CAEX attribute as a list, it references the AttributeType “ListType” or “OrderedListType”, which models a non-ordered list or an ordered list respectively. The child attributes form the list items.

Since CAEX Attributes do not explicitly model the order of the items, the ordered list and the non-ordered list needs to be modelled explicitly.

Normative provisions related to the modelling of lists and arrays are defined in 8.8.

A.2.7.2 Examples

The example in Figure A.42 shows an object *Radio* that stores a list of supported frequencies. The attribute "SupportedFrequencies" references the AML AttributeType "OrderedListType". The child attributes form the list items, each name contains the index of the item within the list. Leading zeros are allowed and exemplarily added for better sortability.

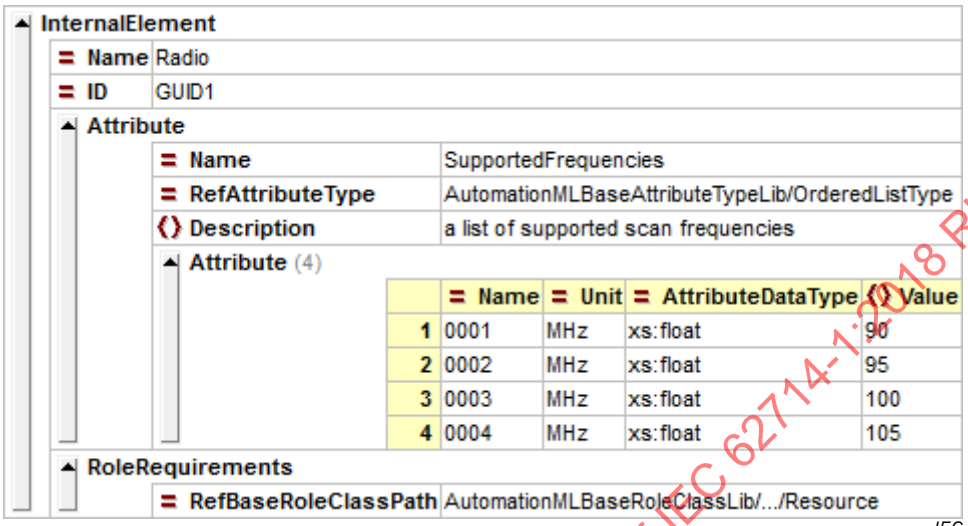


Figure A.42 – Attribute list "SupportedFrequencies"

Figure A.43 shows the related XML code of the example.

```
<InternalElement Name="Radio" ID="GUID1">
  <Attribute Name="SupportedFrequencies" RefAttributeType="AutomationMLBaseAttributeTypeLib/OrderedListType">
    <Description>a list of supported scan frequencies</Description>
    <Attribute Name="0001" Unit="MHz" AttributeDataType="xs:float">
      <Value>90</Value>
    </Attribute>
    <Attribute Name="0002" Unit="MHz" AttributeDataType="xs:float">
      <Value>95</Value>
    </Attribute>
    <Attribute Name="0003" Unit="MHz" AttributeDataType="xs:float">
      <Value>100</Value>
    </Attribute>
    <Attribute Name="0004" Unit="MHz" AttributeDataType="xs:float">
      <Value>105</Value>
    </Attribute>
  </Attribute>
  <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
</InternalElement>
```

IEC

Figure A.43 – XML code for the attribute list "SupportedFrequencies"

The example in Figure A.44 shows the modelling of an array by means of an object "Table" that stores a list of edge points. The attribute "Edges" references the AML AttributeType "ListType". The child attributes form the list items, the names are arbitrary but unique among the siblings. They form again lists, containing the x, y and z positions of the table.

InternalElement

Name	Table
ID	GUID1

Attribute

Name	Edges
RefAttributeType	AutomationMLBaseAttributeTypeLib/ListType
Description	an array of edge points

Attribute (4)

	Name	RefAttributeType	Attribute
1	EdgeNO	AutomationMLBaseAttributeTypeLib/ListType	Attribute (3)
2	EdgeSO	AutomationMLBaseAttributeTypeLib/ListType	Attribute (3)
3	EdgeSW	AutomationMLBaseAttributeTypeLib/ListType	Attribute (3)
4	EdgeNW	AutomationMLBaseAttributeTypeLib/ListType	Attribute (3)

Attribute (3) — EdgeNO

	Name	AttributeDataType	Value
1	x	xs:float	10
2	y	xs:float	10
3	z	xs:float	10

Attribute (3) — EdgeSO

	Name	AttributeDataType	Value
1	x	xs:float	10
2	y	xs:float	0
3	z	xs:float	10

Attribute (3) — EdgeSW

	Name	AttributeDataType	Value
1	x	xs:float	0
2	y	xs:float	0
3	z	xs:float	10

Attribute (3) — EdgeNW

	Name	AttributeDataType	Value
1	x	xs:float	0
2	y	xs:float	10
3	z	xs:float	10

RoleRequirements

RefBaseRoleClassPath	AutomationMLBaseRoleClassLib/.../Resource

IEC

Figure A.44 – Example CAEX model of the array “Edges”

Figure A.45 shows the related XML code of the example.

```
<InternalElement Name="Table" ID="GUID1">
  <Attribute Name="Edges" RefAttributeType="AutomationMLBaseAttributeTypeLib/ListType">
    <Description>an array of edge points</Description>
    <Attribute Name="EdgeNO" RefAttributeType="AutomationMLBaseAttributeTypeLib/ListType">
      <Attribute Name="x" AttributeDataType="xs:float">
        <Value>10</Value>
      </Attribute>
      <Attribute Name="y" AttributeDataType="xs:float">
        <Value>10</Value>
      </Attribute>
      <Attribute Name="z" AttributeDataType="xs:float">
        <Value>10</Value>
      </Attribute>
    </Attribute>
    <Attribute Name="EdgeSO" RefAttributeType="AutomationMLBaseAttributeTypeLib/ListType">
      <Attribute Name="x" AttributeDataType="xs:float">
        <Value>10</Value>
      </Attribute>
      <Attribute Name="y" AttributeDataType="xs:float">
        <Value>0</Value>
      </Attribute>
      <Attribute Name="z" AttributeDataType="xs:float">
        <Value>10</Value>
      </Attribute>
    </Attribute>
    <Attribute Name="EdgeSW" RefAttributeType="AutomationMLBaseAttributeTypeLib/ListType">
      <Attribute Name="x" AttributeDataType="xs:float">
        <Value>0</Value>
      </Attribute>
      <Attribute Name="y" AttributeDataType="xs:float">
        <Value>0</Value>
      </Attribute>
      <Attribute Name="z" AttributeDataType="xs:float">
        <Value>10</Value>
      </Attribute>
    </Attribute>
    <Attribute Name="EdgeNW" RefAttributeType="AutomationMLBaseAttributeTypeLib/ListType">
      <Attribute Name="x" AttributeDataType="xs:float">
        <Value>0</Value>
      </Attribute>
      <Attribute Name="y" AttributeDataType="xs:float">
        <Value>10</Value>
      </Attribute>
      <Attribute Name="z" AttributeDataType="xs:float">
        <Value>10</Value>
      </Attribute>
    </Attribute>
  </Attribute>
  <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
</InternalElement>
```

IEC

Figure A.45 – XML code for the attribute array "Edges"

Annex B
(informative)

XML representation of standard AML base libraries

Figure B.1 below shows the XML text of the standard AML interface class library, role class library and attribute type library.

```
<CAEXFile xmlns="http://www.dke.de/CAEX" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
          instance" SchemaVersion="3.0" FileName="AutomationML2.10BaseLibraries"
          xsi:schemaLocation="http://www.dke.de/CAEX CAEX_ClassModel_V.3.0.xsd">
 <SuperiorStandardVersion>AutomationML 2.10</SuperiorStandardVersion>
 <SourceDocumentInformation OriginID="IEC SC65E WG 9" OriginName="IEC SC65E WG 9"
          OriginVersion="2.10.0" LastWritingDateTime="2016-08-25T09:58:00.0Z"
          OriginProjectID="Automation Markup Language Standard Library"
          OriginRelease="2.10.0" OriginVendor="IEC" OriginVendorURL="www.iec.ch"
          OriginProjectTitle="Automation Markup Language Standard Libraries"/>
 <InterfaceClassLib Name="AutomationMLInterfaceClassLib">
  <Description>Standard Automation Markup Language Interface Class
          Library</Description>
  <Version>2.10.0</Version>
  <InterfaceClass Name="AutomationMLBaseInterface">
   <InterfaceClass Name="Order" RefBaseClassPath="AutomationMLBaseInterface">
     <Attribute Name="Direction" AttributeDataType="xs:string"
       RefAttributeType="AutomationMLBaseAttributeTypeLib/Direction"/>
   </InterfaceClass>
   <InterfaceClass Name="Port" RefBaseClassPath="AutomationMLBaseInterface">
     <Attribute Name="Direction" AttributeDataType="xs:string"
       RefAttributeType="AutomationMLBaseAttributeTypeLib/Direction">
      <Constraint Name="AllowedValues">
       <NominalScaledType>
       <RequiredValue>In</RequiredValue>
       <RequiredValue>Out</RequiredValue>
       <RequiredValue>InOut</RequiredValue>
       </NominalScaledType>
      </Constraint>
     </Attribute>
     <Attribute Name="Cardinality"
       RefAttributeType="AutomationMLBaseAttributeTypeLib/Cardinality">
      <Attribute Name="MinOccur" AttributeDataType="xs:unsignedInt"/>
      <Attribute Name="MaxOccur" AttributeDataType="xs:unsignedInt"/>
     </Attribute>
     <Attribute Name="Category" AttributeDataType="xs:string"
       RefAttributeType="AutomationMLBaseAttributeTypeLib/Category"/>
   </InterfaceClass>
   <InterfaceClass Name="PPRConnector" RefBaseClassPath="AutomationMLBaseInterface"/>
   <InterfaceClass Name="ExternalDataConnector"
       RefBaseClassPath="AutomationMLBaseInterface">
     <Attribute Name="refURI" AttributeDataType="xs:anyURI"
       RefAttributeType="AutomationMLBaseAttributeTypeLib/refURI"/>
     <InterfaceClass Name="COLLADAInterface"
       RefBaseClassPath="ExternalDataConnector"/>
     <InterfaceClass Name="PLCopenXMLInterface"
       RefBaseClassPath="ExternalDataConnector"/>
     <InterfaceClass Name="ExternalDataReference"
       RefBaseClassPath="ExternalDataConnector">
      <Attribute Name="MIMEType" AttributeDataType="xs:string"
       RefAttributeType="AutomationMLBaseAttributeTypeLib/MIMEType"/>
     </InterfaceClass>
   </InterfaceClass>
   <InterfaceClass Name="Communication" RefBaseClassPath="AutomationMLBaseInterface">
     <InterfaceClass Name="SignalInterface" RefBaseClassPath="Communication"/>
   </InterfaceClass>
  </InterfaceClass>
 </InterfaceClassLib>
 <RoleClassLib Name="AutomationMLBaseRoleClassLib">
  <Description>Automation Markup Language base role class library</Description>
  <Version>2.10.0</Version>
  <RoleClass Name="AutomationMLBaseRole">
   <RoleClass Name="Group" RefBaseClassPath="AutomationMLBaseRole">
     <Attribute Name="AssociatedFacet" AttributeDataType="xs:string"
       RefAttributeType="AutomationMLBaseAttributeTypeLib/AssociatedFacet"/>
```

```
    </RoleClass>
    <RoleClass Name="Facet" RefBaseClassPath="AutomationMLBaseRole"/>
    <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="ExternalData" RefBaseClassPath="AutomationMLBaseRole"/>
  </RoleClass>
 </RoleClassLib>
 <AttributeTypeLib Name="AutomationMLBaseAttributeTypeLib">
  <Description>Standard Automation Markup Language Attribute Type Library</Description>
  <Version>2.10.0</Version>
  <AttributeType Name="Direction" AttributeDataType="xs:string">
   <Constraint Name="AllowedValues">
      <NominalScaledType>
       <RequiredValue>In</RequiredValue>
       <RequiredValue>Out</RequiredValue>
       <RequiredValue>InOut</RequiredValue>
      </NominalScaledType>
   </Constraint>
  </AttributeType>
  <AttributeType Name="Cardinality">
   <Attribute Name="MinOccur" AttributeDataType="xs:unsignedInt"/>
   <Attribute Name="MaxOccur" AttributeDataType="xs:unsignedInt"/>
  </AttributeType>
  <AttributeType Name="Category" AttributeDataType="xs:string"/>
  <AttributeType Name="refURI" AttributeDataType="xs:anyURI"/>
  <AttributeType Name="AssociatedFacet" AttributeDataType="xs:string"/>
  <AttributeType Name="ListType"/>
  <AttributeType Name="OrderedListType"/>
  <AttributeType Name="LocalizedAttribute" AttributeDataType="xs:string"/>
  <AttributeType Name="AssociatedExternalValue">
   <Attribute Name="refCAEXAttribute"/>
   <Attribute Name="refURI"
         RefAttributeType="AutomationMLBaseAttributeTypeLib/refURI"/>
   <Attribute Name="Direction"
         RefAttributeType="AutomationMLBaseAttributeTypeLib/Direction"/>
  </AttributeType>
  <AttributeType Name="MIMEType" AttributeDataType="xs:string"/>
  <AttributeType Name="DocLang" AttributeDataType="xs:string"/>
 </AttributeTypeLib>
</CAEXFile>
```

Figure B.1 – XML text of the standard AML interface class library, role class library and attribute type library

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*

IEC 62714-3, *Engineering data exchange format for use in industrial automation systems engineering – Automation markup language – Part 3: Geometry and kinematics*

IEC 62714-4:__[3], *Engineering data exchange format for use in industrial automation systems engineering – Automation Markup Language – Part 4: Logic*

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

Extensible Markup Language (XML) 1.0 1.0:2004, W3C Recommendation [viewed 2017-11-13]. Available at http://www.w3.org/TR/2004/REC-xml-20040204/

3 Under consideration.

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

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

Cette deuxième édition annule et remplace la première édition parue en 2014. Cette édition constitue une révision technique.

Cette édition inclut les modifications techniques majeures suivantes par rapport à l'édition précédente:

a) utilisation de CAEX 3.0 conformément à l'IEC 62424:2016 qui apporte des améliorations techniques, comme les bibliothèques d'attributs, les interfaces imbriquées, les nouveaux champs indiquant la source d'un objet, un affinement du concept de miroir et une prise en charge native de plusieurs rôles, des méta-informations natives relatives à l'outil de source de fichier CAEX, l'identification des instances par l'intermédiaire des ID uniques en lieu et place des chemins, etc.,

b) modélisation améliorée des références aux documents ne relevant pas du domaine d'application de la présente norme,

c) modélisation des références entre les attributs CAEX et les éléments dans des documents externes (dans une feuille Excel, par exemple),

d) bibliothèques de rôles révisées,

e) concept d'accès modifié,

f) modélisation des expressions multilingues,

g) modélisation de listes ou de tableaux d'attributs structuré(e)s,

h) nouveau format de conteneur AML,

i) nouvelle bibliothèque d'attributs AML normalisée.

Le texte de cette Norme internationale est issu des documents suivants:

FDIS	Rapport de vote
65E/582/FDIS	65E/586/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 internationale.

Ce document a été rédigé selon les Directives ISO/IEC, Partie 2.

Une liste de toutes les parties de la série IEC 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 l'IEC.

Le comité a décidé que le contenu de ce document ne sera pas modifié avant la date de stabilité indiquée sur le site web de l'IEC sous "http://webstore.iec.ch" dans les données relatives au document recherché. À cette date, le document sera

- reconduit,
- supprimé,
- remplacé par une édition révisée, ou
- amendé.

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

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

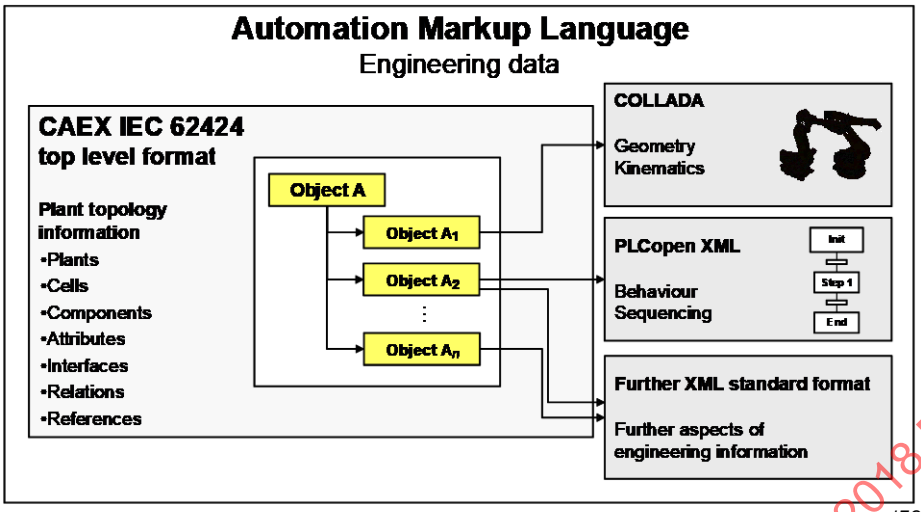
Le format d'échange de données défini dans la série IEC 62714 (Automation Markup Language, AML) est un format de données de type schéma XML pour les données d'ingénierie d'usine. L'AML a été mis au point afin de venir à l'appui de l'échange de données dans un environnement d'outils techniques hétérogène. 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. L'application de l'IEC 62714 dépend du secteur industriel. Elle s'applique à tous les secteurs industriels dont la chaîne d'outils techniques exige un échange de données (dans l'industrie d'assemblage ou l'industrie de transformation, par exemple).

L'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 existant 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.

L'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. CAEX permet d'interconnecter les différents formats de données. Le langage AML a par conséquent une architecture de document répartie intrinsèque.

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



Anglais	Français
Engineering data	Données techniques
CAEX IEC 62424 top level format	Format central CAEX défini dans l'IEC 62424
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	Étape
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 IEC 62714 comporte différentes parties concentrées sur différents aspects:

- IEC 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.

- IEC 62714-2: Bibliothèques de classe de rôles

  Cette partie spécifie d'autres bibliothèques AML.

- IEC 62714-3: Géométrie et cinématique

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

- IEC 62714-4[1]: 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. À défaut, cela ouvrirait la voie à l'utilisation de tout format de données et l'échange de données ne fonctionnerait pas.

L'Article 1 définit le domaine d'application de l'IEC 62714.

L'Article 2 donne les références normatives.

L'Article 3 donne les termes, définitions et abréviations.

L'Article 4 définit la conformité à l'IEC 62714.

L'Article 5 décrit les spécifications générales en matière d'architecture pour l'IEC 62714.

L'Article 6 définit les bibliothèques AML de base.

L'Article 7 décrit la modélisation des données définies par l'utilisateur.

L'Article 8 décrit les concepts AML étendus.

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 l'IEC 62714.

[1] En préparation.

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

1 Domaine d'application

La présente partie de l'IEC 62714 spécifie les exigences générales et l'architecture du langage AML (*Automation Markup Language*) 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 l'IEC 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 cités dans le texte constituent, pour tout ou partie de leur contenu, des exigences du présent document. 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).

IEC 62424:2016, *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*

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

ISO/IEC 29500-2, *Technologies de l'information – Description des documents et langages de traitement – Formats de fichier "Office Open XML" – Partie 2: Conventions de paquetage ouvert*

IETF RFC 2046, *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types* [consulté 2017-11-13]. Adresse <http://www.ietf.org>

IETF RFC 4122, A Universally Unique Identifier (UUID) URN Namespace [consulté 2017-11-13]. Adresse <http://www.ietf.org>

IETF RFC 5646, *Tags for Identifying Languages* [consulté 2017-11-13]. Adresse <http://www.ietf.org>

COLLADA 1.4.1:March 2008, COLLADA – Digital Asset Schema Release 1.4.1 [consulté 2017-11-13]. Adresse <http://www.khronos.org/files/collada_spec_1_4.pdf>

PLCopen XML 2.0:December 3rd 2008 and PLCopen XML 2.0.1:May 8th 2009, *XML formats for IEC 61131-3*
[consulté 2017-11-13]. Adresse <http://www.plcopen.org>

3 Termes, définitions et abréviations

3.1 Termes et définitions

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

L'ISO et l'IEC tiennent à jour des bases de données terminologiques destinées à être utilisées en normalisation, consultables aux adresses suivantes:

- IEC Electropedia: disponible à l'adresse http://www.electropedia.org/
- ISO Online browsing platform: disponible à l'adresse http://www.iso.org/obp

3.1.1
AML
format d'échange de données XML pour les données d'ingénierie d'usine conformément à l'IEC 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 avec un ou plusieurs CAEX RoleRequirements 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 (une classe d'unité de système AML, une classe d'interface AML, une classe de rôle AML ou un type d'attribut AML)

Note 1 à l'article: Les classes AML sont archivées dans des bibliothèques AML. Les classes AML sont de type SystemUnitClass, InterfaceClass, RoleClass ou AttributeType.

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.

Note 4 à l'article: Les classes AML peuvent être définies par l'utilisateur ou être des classes AML normalisées.

3.1.5
attribut AML
attribut CAEX qui appartient à un objet AML et est associé à un attribut défini dans une classe AML ou un AML AttributeType

Note 1 à l'article: Les attributs AML sont décrits comme étant des éléments XML correspondant à l'IEC 62424:2016, A.2.4.

3.1.6
document AML
document AML CAEX spécifique suivant l'IEC 62714 (toutes les parties), 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.

Note 2 à l'article: Les documents AML contiennent des objets AML et/ou des objets définis par l'utilisateur.

Note 3 à l'article: Un document AML peut être composé de plusieurs fichiers, dont la racine est un document AML CAEX.

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

3.1.8
interface AML
point de connexion unique ayant une relation avec une classe d'interface AML

Note 1 à l'article: Les interfaces permettent de décrire des relations entre les objets par la définition de Internal-Links CAEX. Il s'agit, par exemple, d'une interface de signalisation, d'une interface de dispositif ou d'une interface de puissance.

3.1.9
bibliothèque AML
bibliothèque contenant les classes AML

3.1.10
accès AML
interface AML ayant une relation directe ou indirecte avec la classe d'interface AML normalisée Port, permettant de spécifier des interfaces imbriquées

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

3.1.11
groupe AML
objet AML ayant une relation directe ou indirecte avec la classe de rôle AML normalisée Group, offrant une certaine vue des objets AML

3.1.12
facette AML
objet AML ayant une relation directe ou indirecte avec la classe de rôle AML normalisée Facet, offrant une certaine vue des attributs ou interfaces d'un objet AML

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 l'IEC 62424:2016, 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 automatiques 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 l'IEC 62714 (toutes les parties), les GUID sont également présentés sous forme abrégée telle que "GUID1", "GUID2", etc. Cela 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 parente.

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
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.

3.1.20
topologie de l'installation
structure hiérarchique d'une installation, visualisable en tant qu'arborescence d'objets

3.1.21
éditer
modéliser une structure de données d'un document externe destinée à être utilisée avec le format CAEX

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

3.1.22
relation
association d'objets CAEX

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

3.1.23
liaison
connexion entre objets de type ExternalInterface CAEX

Note 1 à l'article: Une liaison est modélisée à l'aide d'InternalLink CAEX.

3.1.24
référence
association entre un objet InternalElement CAEX et les informations à archivage externe

3.2 Abréviations

Les abréviations utilisées dans le présent document figurent au Tableau 1.

Tableau 1 – Abréviations

AML	Automation markup language
CAE	Computer aided engineering (ingénierie assistée par ordinateur)
CAEX	Computer aided engineering eXchange (échange de données techniques assisté par ordinateur)
COLLADA	Collaborative design activity (activité de conception coopérative)
GUID	Global unique identifier (identifiant unique global)
IHM	Interface homme-machine
ID	Identifiant
MES	Manufacturing execution system (système d'exécution de fabrication)
PLC	Automate logique programmable
URL	Uniform resource locator (localisateur de ressources uniformes)
URI	Uniform resource identifier (identificateur de ressources uniformes)
UUID	Universal unique identifier (identifiant unique universel)
XML	eXtensible markup language (langage de balisage extensible)

4 Conformité

Les exigences de l'Article 5, de l'Article 6, de l'Article 7 et de l'Article 8 doivent être satisfaites pour revendiquer la conformité avec la présente partie de l'IEC 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 central CAEX, un format de données neutre conformément à l'IEC 62424:2016, 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 ne sont données qu'à titre indicatif. 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 l'IEC 62424:2016, 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 l'IEC 62424:2016, A.2.8.7.

NOTE 1 Conformément à l'IEC 62424:2016, A.2.8.7, 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 comme identique à l'objet maître. Cela 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 L'IEC 62714 (toutes les parties) 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 dans l'IEC 62424:2016, Annexe D.

Informations concernant les références et les relations: les références et les relations doivent être archivées selon 5.5 et 5.6. Les relations entre les informations à archivage externe doivent être archivées par des mécanismes CAEX. Si cela est exigé, les partenaires de liaison associés doivent être édités dans la description de la topologie de l'installation CAEX, au moyen des ExternalInterfaces CAEX. Ils doivent être déduits des classes d'interface AML normalisées spécifiées en 6.3.

NOTE 3 Les références représentent les liaisons entre les objets CAEX et les informations à archivage externe. Une vue d'ensemble informative concernant les relations est fournie en A.1.7. Les références et l'édition des interfaces sont décrites dans les parties supplémentaires de l'IEC 62714.

NOTE 4 Les relations représentent 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™[2]. Les interfaces COLLADA à interconnecter dans le format central doivent être éditées en tant que ExternalInterfaces CAEX.

NOTE 5 L'IEC 62714 (toutes les parties) 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 spécifiés dans l'IEC 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 (des variables ou des signaux, par exemple) 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. L'IEC 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 spécifiés dans l'IEC 62714-4.

Référencement d'autres formats de données: l'IEC 62714 pourra être étendue par des 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 l'IEC 62714.

---

2 COLLADA est la marque d'un produit fourni par The Khronos Group. Cette information est donnée à l'intention des utilisateurs de la présente Norme et ne signifie nullement que l'IEC approuve ou recommande l'emploi exclusif du produit ainsi désigné. Des produits équivalents peuvent être utilisés s'il est démontré qu'ils conduisent aux mêmes résultats.

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 sous-documents et informations concernant le document AML supérieur

L'IEC 62714 repose sur les formats de document suivants:

- CAEX version 3.0 telle que définie dans l'IEC 62424:2016;
- PLCopenXML 2.0 et 2.0.1;
- COLLADA 1.5.0 tel que spécifié dans l'ISO/PAS 17506 et COLLADA 1.4.1;
- Bibliothèques normalisées AML telles que spécifiées dans la présente partie de l'IEC 62714 et les autres parties de l'IEC 62714.

AML intègre CAEX et fait donc office de norme supérieure.

NOTE 1 Les dispositions normatives concernant les informations sur la version liées aux instances d'objets AML sont définies en 8.7. L'archivage des méta-informations spécifiques aux outils est défini en 5.4.

De fait, les dispositions suivantes s'appliquent:

- Chaque document AML CAEX doit archiver la version AML que suit la présente norme dans l'élément CAEX "SuperiorStandardVersion" conformément à l'IEC 62424:2016, A.2.2.3.
- La valeur de cet élément doit être "AutomationML 2.10" afin de 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 représente le texte XML pour un document CAEX suivant la version AML 2.10.

```
<CAEXFile xmlns="http://www.dke.de/CAEX" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
SchemaVersion="3.0" FileName="AutomationML2.10BaseLibraries" xsi:schemaLocation="
http://www.dke.de/CAEX CAEX_ClassModel_V.3.0.xsd">
  <SuperiorStandardVersion>AutomationML 2.10</SuperiorStandardVersion>
```

IEC

Figure 2 – Informations concernant les versions de documents AML

- Chaque bibliothèque normalisée 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 l'IEC 62714.
- Si cela est exigé, 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 AML ne sont pas définies dans la présente partie de l'IEC 62714.
- Il est interdit d'archiver les mêmes bibliothèques de différentes versions dans le même fichier AML.

  NOTE 2 Cela 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.

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:

- Conformément à l'IEC 62424:2016, chaque document AML doit fournir des informations concernant l'outil source 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. Cela 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.
- Le présent document recommande d'utiliser l'élément CAEX facultatif SourceObjectInformation avec ses attributs OriginID et SourceObjID conformément à l'IEC 62424:2016, A.2.2.7, afin d'identifier l'outil source de chaque instance d'objet AML (InternalElement, ExternalInterface).

La Figure 3 présente le texte XML exigé des informations d'origine exigées du document. L'exemple indique les informations sources des bibliothèques normalisées fournies avec la présente partie de l'IEC 62714.

```
<SourceDocumentInformation OriginID="IEC SC65E WG 9" OriginName="IEC SC65E WG 9"
OriginVersion="2.10.0" LastWritingDateTime="2016-08-25T09:58:00.0Z" OriginProjectID="Automation
Markup Language Standard Library" OriginRelease="2.10.0" OriginVendor="IEC" OriginVendorURL="
www.iec.ch" OriginProjectTitle="Automation Markup Language Standard Libraries"/>
```

IEC

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

5.5 Spécification des relations AML

5.5.1 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 l'IEC 62714 distingue deux mécanismes d'archivage de ces informations: les références et les relations. Le Paragraphe 5.5 porte sur les relations, alors que le Paragraphe 5.6 porte sur les références. Une vue d'ensemble informative concernant les relations et les références est fournie en A.1.7.

5.5.2 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 d'une InstanceHierarchy CAEX ou d'une SystemUnitClass CAEX.
- Un objet AML peut être un singleton sans relation avec une SystemUnitClass.

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

  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.

- Conformément à l'IEC 62424:2016, A.2.2.7, il convient que les modifications 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". De plus, dans l'ancienne classe, il convient d'archiver le chemin de la nouvelle version dans la balise CAEX "NewVersion".

  NOTE 3 Cette disposition prend en charge le suivi des modifications opérées dans les différentes versions d'une classe.

5.5.3 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:

- Il convient de déduire les ExternalInterfaces directement ou indirectement à partir d'une classe d'interface AML normalisée.

  NOTE 1 La bibliothèque de classes d'interfaces AML normalisées est spécifiée en 6.3. La classe d'interface AML définit 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.

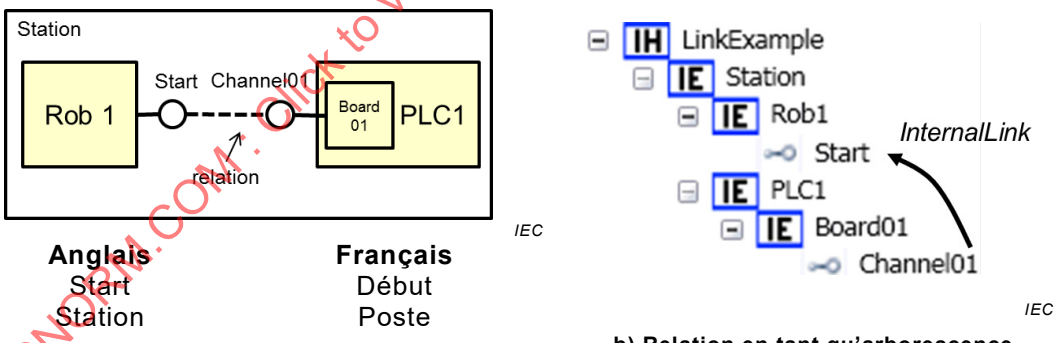
- Les documents COLLADA peuvent être interconnectés. Les interfaces COLLADA correspondantes peuvent être tous les éléments ayant un URI valide. Si l'interconnexion de ces nœuds au format CAEX est exigée, lesdits nœuds doivent être édités dans ce format en ajoutant une ExternalInterface CAEX à l'objet correspondant. Cette ExternalInterface doit être déduite de la classe d'interface AML normalisée "COLLADAInterface" ou de l'une de ses dérivées.

  NOTE 2 La classe d'interface normalisée "COLLADAInterface" est spécifiée en 6.3.7. Les détails sont spécifiés dans l'IEC 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 exigée, lesdits éléments doivent être édités en ajoutant une ExternalInterface CAEX à l'objet correspondant. Cette ExternalInterface doit être déduite de la classe d'interface AML normalisée "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 spécifiés dans l'IEC 62714-4.

La Figure 4a) présente un exemple comprenant un robot "Rob1" et un automate PLC "PLC1", chacun d'entre eux ayant une interface de signal connectée. La Figure 4b) présente cet exemple comme une hiérarchie d'objet.



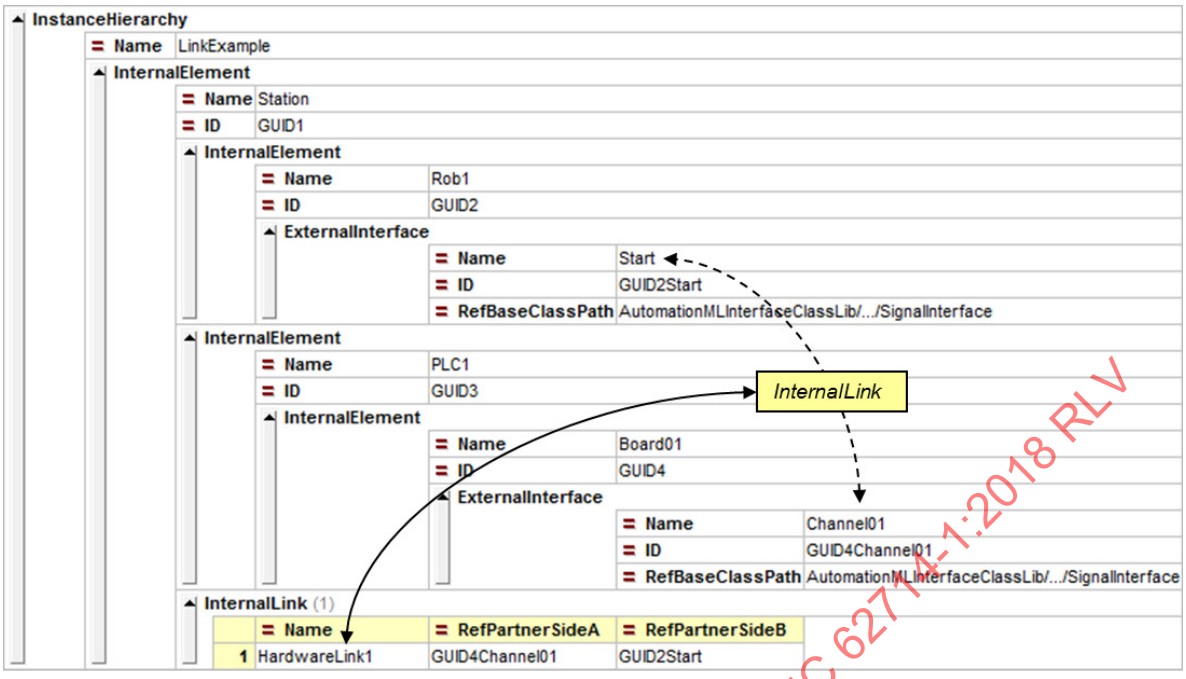
a) Relation en tant que schéma de principe

b) Relation en tant qu'arborescence d'objet

Figure 4 – Exemple de relation en tant que schéma de principe et en tant qu'arborescence d'objet

La Figure 5 et la Figure 6 représentent la représentation AML de l'exemple donné. Le texte XML complet de l'élément InstanceHierarchy pour cet exemple, comprenant tous les InternalElements "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 (voir la Figure 4) afin d'améliorer la lisibilité.



IEC

Figure 5 – Exemple de relation entre les objets "PLC1" et "Rob1"

```
<InstanceHierarchy Name="LinkExample">
  <InternalElement Name="Station" ID="GUID1">
    <InternalElement Name="Rob1" ID="GUID2">
      <ExternalInterface Name="Start" ID="GUID2Start" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface"/>
    </InternalElement>
    <InternalElement Name="PLC1" ID="GUID3">
      <InternalElement Name="Board01" ID="GUID4">
        <ExternalInterface Name="Channel01" ID="GUID4Channel01" RefBaseClassPath="AutomationMLInterfaceClassLib/.../SignalInterface"/>
      </InternalElement>
    </InternalElement>
    <InternalLink Name="HardwareLink1" RefPartnerSideA="GUID4Channel01" RefPartnerSideB="GUID2Start"/>
  </InternalElement>
</InstanceHierarchy>
```

IEC

Figure 6 – Texte XML de l'exemple de relation entre les objets "PLC1" et "Rob1"

5.5.4 Identification des objets

Le présent document recommande l'identification des InternalElements et ExternalInterfaces au moyen de GUID conformément au RFC 4122. Pour comparer deux identifiants, les dispositions suivantes s'appliquent:

- Si l'identifiant d'un InternalElement ou d'une ExternalInterface est un GUID, les deux identifiants doivent être identiques si la valeur numérique contenue l'est également. Les crochets, les accolades, les traits d'union ou les espaces sont admis, mais ils doivent être non pertinents.

  Exemple sans accolade: 48d23207-09e0-4104-82fb-344007d2b7f5

  Exemple avec accolade: { 48d23207-09e0-4104-82fb-344007d2b7f5 }

5.6 Spécification de référence de document AML

5.6.1 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 normalisée "ExternalDataConnector" ou l'une de ses dérivées.

5.6.2 Référencement de documents COLLADA

Le référencement des documents COLLADA doit être effectué sur la base de la classe d'interface AML normalisée "COLLADAInterface" ou de l'une de ses dérivées. Cette classe est spécifiée en 6.3.7. Les détails sont spécifiés dans l'IEC 62714-3.

5.6.3 Référencement de documents XML PLCopen

Le référencement des documents XML PLCopen doit être effectué sur la base de l'interface AML normalisée "PLCopenXMLInterface" ou de l'une de ses dérivées. Cette classe est spécifiée en 6.3.8. Les détails sont spécifiés dans l'IEC 62714-4.

5.6.4 Référencement de documents supplémentaires dans le domaine d'application de l'IEC 62714 (toutes les parties)

Les extensions futures de l'IEC 62714 peuvent ajouter des types d'interfaces supplémentaires pour le référencement de types de documents supplémentaires. Elles sont spécifiées dans des parties distinctes de l'IEC 62714 et non dans le domaine d'application de la présente partie de la série. Les dispositions suivantes s'appliquent concernant ces extensions:

- Si des types de documents supplémentaires doivent être ajoutés à l'IEC 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 normalisée "ExternalDataConnector".
- Il convient d'archiver les références en utilisant les mêmes attributs normalisés fournis par les classes d'interface normalisées.

5.6.5 Référencement de documents ne relevant pas du domaine d'application de l'IEC 62714 (toutes les parties)

S'il s'avère nécessaire qu'un document externe ne relevant pas du domaine d'application de la présente norme soit référencé par le document AML (des manuels, des instructions ou des résultats d'études spécifiques comme des programmes de commande natifs, par exemple), les dispositions suivantes s'appliquent:

- Un document ne relevant pas du domaine d'application de l'IEC 62714 doit être modélisé au moyen d'un InternalElement CAEX avec une association directe ou indirecte au RoleClass "ExternalData" défini en 6.4.12. Le RoleClass référencé doit spécifier le contenu du document. Plusieurs rôles avec le même contenu peuvent être attribués à un document.

  NOTE 1 Chaque document peut contenir plusieurs types de contenu (une nomenclature et un manuel d'utilisateur, par exemple).

  NOTE 2 Si nécessaire, chaque document peut faire référence à plusieurs fichiers (s'il est divisé en fichiers différents, par exemple).
- Si un document est spécifique à une langue, il doit contenir un attribut CAEX de type "DocLang". Si un document contient plusieurs langues, il doit contenir une liste d'attributs non triés (voir 8.8) avec des attributs de type "DocLang".
- Chaque document doit contenir au moins une ExternalInterface qui doit être déduite directement ou indirectement de la classe d'interface "ExternalDataReference".
- Cette ExternalInterface doit modéliser l'URI vers le document externe au moyen de l'attribut CAEX prédéfini de type "refURI" hérité de la classe d'interface AML normalisée "ExternalDataConnector", et doit en outre modéliser le type du document à l'aide de l'attribut CAEX prédéfini "MIMEType" du type “MIMEType” hérité de la classe d'interface AML normalisée "ExternalDataReference".

Des informations supplémentaires et un exemple sont fournis en A.1.5.

5.6.6 Référencement des attributs CAEX avec des éléments dans les documents externes

S'il s'avère nécessaire d'associer un attribut CAEX à un élément connexe d'un document externe (un document XML externe ou un document Excel ne relevant pas du domaine d'application de la présente norme, par exemple), les dispositions suivantes s'appliquent:

- Chaque référence entre un attribut CAEX et un élément d'un document externe doit être modélisée par une ExternalInterface CAEX conformément aux dispositions spécifiées en 5.6.5.
- Pour chaque référence entre un attribut CAEX et un élément d'un document externe, cette ExternalInterface CAEX modélise un attribut supplémentaire de type "AssociatedExternalValue" contenant des attributs imbriqués.
- Le premier attribut imbriqué doit refléter l'attribut CAEX. Cela signifie que le GUID de l'objet parent des attributs, séparé par un "/" et le nom de l'attribut, est modélisé dans l'attribut CAEX "RefAttributeType". Le nom de cet attribut doit être non pertinent, mais unique parmi ses objets jumeaux.
- Le deuxième attribut imbriqué de type "refURI" doit référencer l'élément dans le document externe. Cette référence doit être le même document ou un sous-document du document externe référencé dans le InternalElement parent (voir 5.6.5). La syntaxe de cette référence ne relève pas du domaine d'application de la présente norme et exige un élément de document externe référençable.
- Le troisième attribut imbriqué de type "Direction" doit modéliser le sens du flux d'informations. La valeur d'attribut doit être "In" si l'attribut externe est utilisé par l'attribut CAEX (il s'agit alors d'un attribut externe de début) ou doit être "Out" si l'attribut CAEX est un attribut de début et que l'élément externe utilise la valeur de l'attribut CAEX. La valeur "InOut" est interdite.

Des informations supplémentaires et un exemple sont fournis en A.1.6.

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 AML normalisée et peuvent être supprimés après l'instanciation lorsqu'ils ne sont pas nécessaires.

NOTE Les bibliothèques spécifiques au domaine relèvent du domaine d'application des autres parties de l'IEC 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 normalisée "AutomationMLBaseRole".
- Toutes les interfaces AML doivent être associées directement ou indirectement à la classe d'interface normalisée "AutomationMLBaseInterface".

6.3 Bibliothèque de classes d'interface AML – AutomationMLInterfaceClassLib

6.3.1 Généralités

La bibliothèque "AutomationMLInterfaceClassLib" suivante est modélisée conformément à l'IEC 62424:2016, Article 7, Annexe A et Annexe C. L'IEC 62714 (toutes les parties) utilise le concept d'interface CAEX. Les extensions définies par l'utilisateur de cette bibliothèque AML sont admises comme cela est spécifié en 7.4.

Chaque interface doit être déduite directement ou indirectement d'une classe de la bibliothèque "AutomationMLInterfaceClassLib" normalisée suivante, selon le Tableau 2. Les Paragraphes 6.3.2 à 6.3.11 spécifient les classes d'interfaces de manière détaillée.

Tableau 2 – Classes d'interfaces de la bibliothèque AutomationMLInterfaceClassLib

Bibliothèque des classes AML InterfaceClass	InterfaceClass	Description
AutomationMLInterfaceClassLib AutomationMLBaseInterface Order Port PPRConnector ExternalDataConnector COLLADAInterface PLCopenXMLInterface ExternalDataReference Communication SignalInterface	AutomationMLBaseInterface	Type d'interface abstraite
	Order	Interface de description des classements
	Port	Interface de description et d'interconnexion des accès
	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
	ExternalDataReference	Interface avec des documents externes ne relevant pas du domaine d'application de la présente norme
	Communication	Interface de communication générique
	SignalInterface	Interface de signal générique

La Figure 7 présente une vue de tableau et la Figure 8 présente le texte XML de la bibliothèque InterfaceClassLib de type AML normalisée. Les Paragraphes 6.3.2 à 6.3.10 fournissent des informations détaillées concernant les classes.

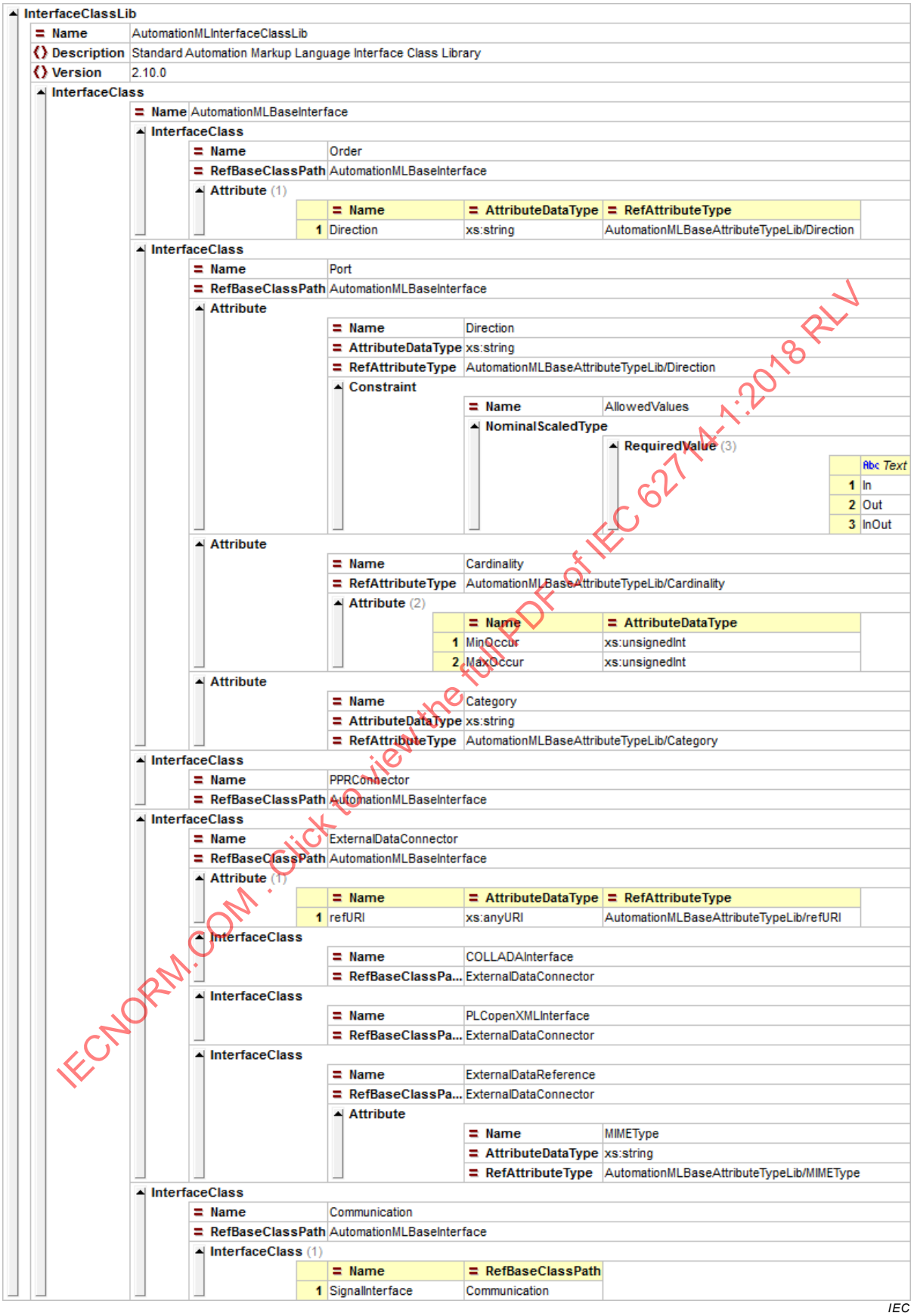


Figure 7 – Bibliothèque de classes d'interfaces de type AML

```
<InterfaceClassLib Name="AutomationMLInterfaceClassLib">
  <Description>Standard Automation Markup Language Interface Class Library</Description>
  <Version>2.10.0</Version>
  <InterfaceClass Name="AutomationMLBaseInterface">
    <InterfaceClass Name="Order" RefBaseClassPath="AutomationMLBaseInterface">
      <Attribute Name="Direction" AttributeDataType="xs:string" RefAttributeType="AutomationMLBaseAttributeTypeLib/Direction"/>
    </InterfaceClass>
    <InterfaceClass Name="Port" RefBaseClassPath="AutomationMLBaseInterface">
      <Attribute Name="Direction" AttributeDataType="xs:string" RefAttributeType="AutomationMLBaseAttributeTypeLib/Direction">
        <Constraint Name="AllowedValues">
          <NominalScaledType>
            <RequiredValue>In</RequiredValue>
            <RequiredValue>Out</RequiredValue>
            <RequiredValue>InOut</RequiredValue>
          </NominalScaledType>
        </Constraint>
      </Attribute>
      <Attribute Name="Cardinality" RefAttributeType="AutomationMLBaseAttributeTypeLib/Cardinality">
        <Attribute Name="MinOccur" AttributeDataType="xs:unsignedInt"/>
        <Attribute Name="MaxOccur" AttributeDataType="xs:unsignedInt"/>
      </Attribute>
      <Attribute Name="Category" AttributeDataType="xs:string" RefAttributeType="AutomationMLBaseAttributeTypeLib/Category"/>
    </InterfaceClass>
    <InterfaceClass Name="PPRConnector" RefBaseClassPath="AutomationMLBaseInterface"/>
    <InterfaceClass Name="ExternalDataConnector" RefBaseClassPath="AutomationMLBaseInterface">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI" RefAttributeType="AutomationMLBaseAttributeTypeLib/refURI"/>
      <InterfaceClass Name="COLLADAInterface" RefBaseClassPath="ExternalDataConnector"/>
      <InterfaceClass Name="PLCopenXMLInterface" RefBaseClassPath="ExternalDataConnector"/>
      <InterfaceClass Name="ExternalDataReference" RefBaseClassPath="ExternalDataConnector">
        <Attribute Name="MIMEType" AttributeDataType="xs:string" RefAttributeType="AutomationMLBaseAttributeTypeLib/MIMEType"/>
      </InterfaceClass>
    </InterfaceClass>
    <InterfaceClass Name="Communication" RefBaseClassPath="AutomationMLBaseInterface">
      <InterfaceClass Name="SignalInterface" RefBaseClassPath="Communication"/>
    </InterfaceClass>
  </InterfaceClass>
</InterfaceClassLib>
```

IEC

Figure 8 – Description XML de la bibliothèque de classes d'interfaces de type AML

6.3.2 Bibliothèque InterfaceClass AutomationMLBaseInterface

Le Tableau 3 spécifie la classe d'interface "AutomationMLBaseInterface".

Tableau 3 – 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 parente pour la description de toutes les classes d'interfaces AML.
Classe parente	Aucune
Chemin de l'élément de référence	AutomationMLInterfaceClassLib/AutomationMLBaseInterface
Attributs	Aucune

6.3.3 InterfaceClass Order

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

Tableau 4 – 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 (un successeur ou un prédécesseur, par exemple).
Classe parente	AutomationMLInterfaceClassLib/AutomationMLBaseInterface
Chemin de l'élément de référence	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Order
Attributs	Nom: Direction RefAttributeType: AutomationMLBaseAttributeTypeLib/Direction Sémantique: voir 6.5.2

6.3.4 InterfaceClass Port

Le Tableau 5 spécifie la classe de rôle "Port".

Tableau 5 – Attributs facultatifs des interfaces Port AML

Nom de classe	Port
Description	La classe d'interface "Port" est un type d'interface pour les interfaces qui regroupe de nombreuses interfaces imbriquées et permet de décrire les interfaces complexes de cette manière. Les interfaces Port AML doivent référencer cette classe d'interface. Les détails et les exemples sont spécifiés en 8.2.
Classe parente	AutomationMLInterfaceClassLib/AutomationMLBaseInterface
Chemin de l'élément de référence	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Port
Attributs	Nom: Direction RefAttributeType: AutomationMLBaseAttributeTypeLib/Direction Sémantique: voir 6.5.2
	Nom: Cardinality RefAttributeType: AutomationMLBaseAttributeTypeLib/Cardinality Sémantique: voir 6.5.2
	Nom: Catégorie RefAttributeType: AutomationMLBaseAttributeTypeLib/Category Sémantique: voir 6.5.2

6.3.5 InterfaceClass PPRConnector

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

Tableau 6 – 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.2.5 pour de plus amples informations.
Classe parente	AutomationMLInterfaceClassLib/AutomationMLBaseInterface
Chemin de l'élément de référence	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/PPRConnector
Attributs	

6.3.6 InterfaceClass ExternalDataConnector

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

Tableau 7 – 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
Chemin de l'élément de référence	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector
Attributs	Nom: refURI RefAttributeType: AutomationMLBaseAttributeTypeLib/refURI Sémantique: voir 6.5.2

6.3.7 InterfaceClass COLLADAInterface

Le Tableau 8 spécifie la classe d'interface "COLLADAInterface". Les détails sont spécifiés dans l'IEC 62714-3.

Tableau 8 – 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 spécifiés dans l'IEC 62714-3.
Classe parente	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector
Chemin de l'élément de référence	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ ExternalDataConnector/COLLADAInterface
Attributs	Aucun

6.3.8 InterfaceClass PLCopenXMLInterface

Le Tableau 9 spécifie la classe d'interface "PLCopenXMLInterface". Les détails sont spécifiés dans l'IEC 62714-4.

Tableau 9 – 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 spécifiés dans l'IEC 62714-4.
Classe parente	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector
Chemin de l'élément de référence	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ ExternalDataConnector/PLCopenXMLInterface
Attributs	Aucun

6.3.9 InterfaceClass ExternalDataReference

Le Tableau 10 spécifie la classe d'interface "ExternalDataReference". Les détails sont spécifiés en 5.6.5.

Tableau 10 – InterfaceClass ExternalDataReference

Nom de classe	ExternalDataReference
Description	La classe d'interface "ExternalDataReference" doit permettre de référencer des documents externes ne relevant pas du domaine d'application d'AML. Les détails sont spécifiés en 5.6.5.
Classe parente	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector
Chemin de l'élément de référence	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ ExternalDataConnector/ExternalDataReference
Attributs	Nom: MIMEType RefAttributeType: AutomationMLBaseAttributeTypeLib/MIMEType Sémantique: voir 6.5.2

6.3.10 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 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 parente	AutomationMLInterfaceClassLib/AutomationMLBaseInterface
Chemin de l'élément de référence	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication
Attributs	Aucun

6.3.11 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 4.
Classe parente	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication
Chemin de l'élément de référence	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication/ SignalInterface
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 normalisées exigée 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.

Un rôle peut être associé à une SystemUnitClass au moyen de sa/ses SupportedRoleClass(es) CAEX. Cela indique que la classe est en mesure de prendre en charge le ou les rôles référencés. Plusieurs SupportedRoleClasses sont pris en charge. Le MappingObject assure le mapping entre les attributs et interfaces du rôle et les attributs et interfaces SystemUnitClass. Dès qu'une SystemUnitClass est instanciée, l'InternalElement connexe contient ces informations. Toutefois, cela n'indique pas que l'instance joue réellement un rôle dans son contexte individuel.

Les RoleRequirements CAEX modélisent les rôles réels ou demandés d'un InternalElement CAEX. Plusieurs RoleRequirements sont pris en charge. Le ou les rôles réels sont référencés et les exigences de l'instance individuelle concernant le rôle sont modélisées dans la RoleRequirement. Le MappingObject permet de mapper les attributs et les interfaces entre la classe de rôle et l'InternalElement, si cela est exigé.

Tout en associant une classe de rôle à un objet AML, cet objet AML comporte une sémantique. "Resource" et "Robot" sont des exemples de classe de rôle. Des bibliothèques supplémentaires sont décrites dans l'IEC 62714-2. Tous les attributs décrits font partie intégrante de la bibliothèque AML normalisée et peuvent être supprimés après l'instanciation 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. La Figure 9 à la Figure 11 présentent la RoleClass de base normalisée comme arborescence d'objet, table XML et texte XML. Les détails de chaque classe de rôle sont donnés de 6.4.2 à 6.4.12.

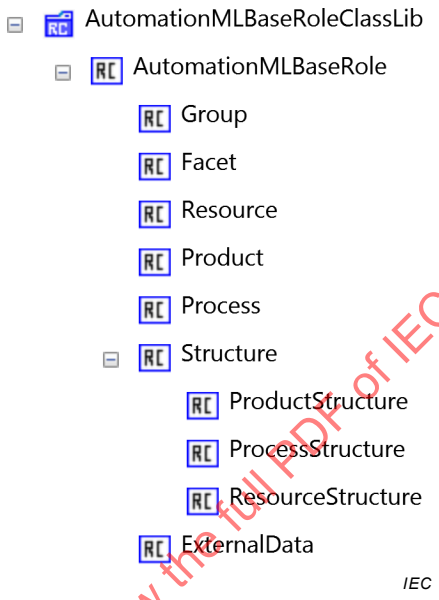


Figure 9 – Bibliothèque de classes de rôles de type AML

RoleClassLib

Name	AutomationMLBaseRoleClassLib
Description	Automation Markup Language base role class library
Version	2.10.0

RoleClass

Name	AutomationMLBaseRole

RoleClass

Name	Group
RefBaseClassPath	AutomationMLBaseRole

Attribute

Name	AssociatedFacet
AttributeDataType	xs:string
RefAttributeType	AutomationMLBaseAttributeTypeLib/AssociatedFacet

RoleClass

Name	Facet
RefBaseClassPath	AutomationMLBaseRole

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	Structure
2	ProcessStructure	Structure
3	ResourceStructure	Structure

RoleClass

Name	ExternalData
RefBaseClassPath	AutomationMLBaseRole

IEC

Figure 10 – AutomationMLBaseRoleClassLib

```
<RoleClassLib Name="AutomationMLBaseRoleClassLib">
  <Description>Automation Markup Language base role class library</Description>
  <Version>2.10.0</Version>
  <RoleClass Name="AutomationMLBaseRole">
    <RoleClass Name="Group" RefBaseClassPath="AutomationMLBaseRole">
      <Attribute Name="AssociatedFacet" AttributeDataType="xs:string" RefAttributeType="AutomationMLBaseAttributeTypeLib/AssociatedFacet"/>
    </RoleClass>
    <RoleClass Name="Facet" RefBaseClassPath="AutomationMLBaseRole"/>
    <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="ExternalData" RefBaseClassPath="AutomationMLBaseRole"/>
  </RoleClass>
</RoleClassLib>
```

IEC

Figure 11 – Texte XML de l'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 normalisées ou définies par l'utilisateur.
Classe parente	Aucun
Chemin de l'élément de référence	AutomationMLBaseRoleClassLib/AutomationMLBaseRole
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 parente	AutomationMLBaseRoleClassLib/AutomationMLBaseRole
Chemin de l'élément de référence	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Group
Attributs	Nom: AssociatedFacet RefAttributeType: AutomationMLBaseAttributeTypeLib/AssociatedFacet Sémantique: voir 6.5.2

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 parente	AutomationMLBaseRoleClassLib/AutomationMLBaseRole
Chemin de l'élément de référence	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Facet
Attributs	Aucun

6.4.5 RoleClass Resource

Le Tableau 16 spécifie la classe de rôle "Resource".

Tableau 16 – RoleClass Resource

Nom de classe	Plan d'affectation
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.2.5.
Classe parente	AutomationMLBaseRoleClassLib/AutomationMLBaseRole
Chemin de l'élément de référence	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource
Attributs	Aucun

De plus, si cela est exigé, 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).

6.4.6 RoleClass Product

Le Tableau 17 spécifie la classe de rôle "Product".

Tableau 17 – 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.2.5.
Classe parente	AutomationMLBaseRoleClassLib/AutomationMLBaseRole
Chemin de l'élément de référence	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Product
Attributs	Aucun

De plus, si cela est exigé, 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).

6.4.7 RoleClass Process

Le Tableau 18 spécifie la classe de rôle "Process".

Tableau 18 – 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.2.5.
Classe parente	AutomationMLBaseRoleClassLib/AutomationMLBaseRole
Chemin de l'élément de référence	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Process
Attributs	Aucun

De plus, si cela est exigé, 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).

6.4.8 RoleClass Structure

Le Tableau 19 spécifie la classe de rôle "Structure".

Tableau 19 – 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 parente	AutomationMLBaseRoleClassLib/AutomationMLBaseRole
Chemin de l'élément de référence	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure
Attributs	Aucun

6.4.9 RoleClass ProductStructure

Le Tableau 20 spécifie la classe de rôle "ProductStructure".

Tableau 20 – 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 parente	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure
Chemin de l'élément de référence	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure/ ProductStructure
Attributs	Aucun

6.4.10 RoleClass ProcessStructure

Le Tableau 21 spécifie la classe de rôle "ProcessStructure".

Tableau 21 – 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
Chemin de l'élément de référence	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure/ ProcessStructure
Attributs	Aucun

6.4.11 RoleClass ResourceStructure

Le Tableau 22 spécifie la classe de rôle "ResourceStructure".

Tableau 22 – 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 parente	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure
Chemin de l'élément de référence	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure/ ResourceStructure
Attributs	Aucun

6.4.12 RoleClass ExternalData

Le Tableau 23 spécifie la classe de rôle "ExternalData".