



# <AutomationML/>

**The Glue for Seamless  
Automation Engineering**

**Whitepaper AutomationML Edition 2.1  
Part 1 – Architecture and General  
Requirements**

**State: July 2018**

© AutomationML consortium

Version 2.1.0, July 2018

**Contact:** [office@automationml.org](mailto:office@automationml.org)

**Table of content**

Table of content.....	3
List of figures .....	6
List of tables .....	8
1 Introduction .....	9
2 Scope .....	10
3 Normative references .....	10
4 Terms, definitions, and abbreviations .....	11
4.1 Terms and definitions.....	11
4.1.1 AML .....	11
4.1.2 Automation object.....	11
4.1.3 AML object .....	11
4.1.4 AML class.....	11
4.1.5 AML attribute .....	11
4.1.6 AML document .....	11
4.1.7 AML file .....	11
4.1.8 AML interface .....	11
4.1.9 AML library .....	11
4.1.10 AML Port .....	11
4.1.11 AML Group .....	12
4.1.12 AML Facet .....	12
4.1.13 CAEX .....	12
4.1.14 Copy-instance-relation .....	12
4.1.15 Universal unique identifier .....	12
4.1.16 Global unique identifier .....	12
4.1.17 Inheritance relation.....	12
4.1.18 Instance .....	12
4.1.19 Topology.....	12
4.1.20 Plant topology.....	12
4.1.21 Publish, verb.....	13
4.1.22 Relation .....	13
4.1.23 Link .....	13
4.1.24 Reference.....	13
4.2 Abbreviations .....	13
5 AML architecture specifications .....	14
5.1 General .....	14
5.2 General AML architecture .....	14
5.3 Sub document versions and AML superior document information .....	15
5.4 Meta information about the AML source tool .....	16
5.5 AML relations specification .....	16
5.5.1 General.....	16
5.5.2 Class-instance-relations.....	16
5.5.3 Instance-instance-relations .....	17
5.6 AML document reference specification.....	19
5.6.1 General.....	19
5.6.2 Referencing COLLADA documents .....	19

5.6.3	Referencing PLCOpen XML documents.....	19
5.6.4	Referencing additional documents in the scope of IEC62714 .....	19
5.6.5	Referencing documents outside of the scope of IEC62714.....	19
5.6.6	Referencing CAEX attributes to items in external documents .....	20
6	AML base libraries .....	21
6.1	General .....	21
6.2	General provisions .....	21
6.3	AML interface class library – AutomationMLInterfaceClassLib .....	21
6.3.1	General.....	21
6.3.2	InterfaceClass AutomationMLBaseInterface.....	23
6.3.3	InterfaceClass Order .....	23
6.3.4	InterfaceClass Port.....	24
6.3.5	InterfaceClass PPRConnector .....	24
6.3.6	InterfaceClass ExternalDataConnector.....	25
6.3.7	InterfaceClass COLLADAInterface .....	25
6.3.8	InterfaceClass PLCOpenXMLInterface.....	26
6.3.9	InterfaceClass ExternalDataReference.....	26
6.3.10	InterfaceClass Communication .....	26
6.3.11	InterfaceClass SignalInterface .....	27
6.4	AML basic role class library – AutomationMLBaseRoleClassLib .....	28
6.4.1	General.....	28
6.4.2	RoleClass AutomationMLBaseRole .....	29
6.4.3	RoleClass Group .....	30
6.4.4	RoleClass Facet.....	30
6.4.5	RoleClass Resource .....	31
6.4.6	RoleClass Product.....	31
6.4.7	RoleClass Process .....	31
6.4.8	RoleClass Structure .....	32
6.4.9	RoleClass ProductStructure.....	32
6.4.10	RoleClass ProcessStructure .....	33
6.4.11	RoleClass ResourceStructure.....	33
6.4.12	RoleClass ExternalData .....	33
6.5	AML basic attribute type library .....	34
6.5.1	General.....	34
6.5.2	AML basic attribute type library AutomationMLBaseAttributeTypeLib.....	34
7	Modelling of user-defined data.....	39
7.1	General .....	39
7.2	User-defined attributes.....	39
7.3	User defined AttributeTypes .....	39
7.4	User-defined InterfaceClasses .....	40
7.5	User-defined RoleClasses .....	41
7.6	User-defined SystemUnitClasses .....	42
7.7	User-defined InstanceHierarchies .....	43
8	Extended AML concepts .....	44
8.1	General overview .....	44
8.2	AML Port interface .....	44

8.3	AML Facet object .....	44
8.4	AML Group object .....	44
8.5	Splitting of AML top-level data into different documents .....	45
8.6	Internationalization, AML multilingual expression .....	45
8.7	Version information of AML objects .....	45
8.8	Modelling of structured attribute lists or arrays .....	45
8.9	AML Container .....	46
A	General introduction into the Automation Markup Language .....	48
A.1	General Automation Markup Language concepts .....	48
A.1.1	The Automation Markup Language architecture .....	48
A.1.2	Modelling of plant topology information .....	49
A.1.3	Referencing geometry and kinematics information .....	51
A.1.4	Referencing logic information .....	51
A.1.5	Referencing documents outside of the scope of IEC62714 .....	51
A.1.6	Interlinking CAEX attributes and attributes in external documents .....	52
A.1.7	Modelling of relations .....	53
A.2	Extended AML concepts and examples .....	57
A.2.1	General overview .....	57
A.2.2	AML Port concept .....	57
A.2.3	AML Facet concept .....	60
A.2.4	AML Group concept .....	62
A.2.5	Process-Product-Resource concept .....	66
A.2.6	AML multilingual expression concept .....	74
A.2.7	Attribute lists and arrays .....	75
B.	XML Representation of standard AML base libraries .....	79
	Bibliography .....	80

**List of figures**

Figure 1.1 – Overview of the engineering data exchange format AML .....	9
Figure 5.1 - AML document version information.....	15
Figure 5.2 - XML text of the AML source tool information .....	16
Figure 5.3 - Example of a relation as block diagram and as object tree .....	17
Figure 5.4 - Example relation between the objects “PLC1” and “Rob1” .....	18
Figure 5.5 - XML text of the example relation between the objects “PLC1” and “Rob1” .....	18
Figure 6.1 – AML basic interface class library .....	22
Figure 6.2 – XML description of the AML basic interface class library .....	23
Figure 6.3 - AML basic role class library.....	28
Figure 6.4 - AutomationMLBaseRoleClassLib.....	29
Figure 6.5 – XML text of the AutomationMLBaseRoleClassLib.....	29
Figure 6.6 – AML basic attribute type library .....	37
Figure 6.7 – XML text of the AutomationMLBaseAttributeTypeLib.....	38
Figure 7.1 – Example of a user-defined attribute .....	39
Figure 7.2 – Examples for user-defined AttributeTypes .....	40
Figure 7.3 – XML code of the examples for user-defined AttributeTypes .....	40
Figure 7.4 – Example of a user-defined InterfaceClass in a user-defined InterfaceClassLib .....	41
Figure 7.5 – XML code of the Example of a user-defined InterfaceClass in a user-defined InterfaceClassLib .....	41
Figure 7.6 – Example of a user-defined RoleClass in a user-defined RoleClassLib.....	42
Figure 7.7 – XML code of the example of a user-defined RoleClass in a user-defined RoleClassLib .....	42
Figure 7.8 – Examples for different user-defined SystemUnitClasses .....	42
Figure 7.9 – XML code of the examples for different user-defined SystemUnitClasses .....	42
Figure 7.10 – Example of a user-defined InstanceHierarchy .....	43
Figure 7.11 – AML representation of a user-defined InstanceHierarchy .....	43
Figure A.1 - AML general architecture .....	48
Figure A.2 - Plant topology with AML .....	50
Figure A.3 - Reference from CAEX to a COLLADA document.....	51
Figure A.4 – Reference from a CAEX to a PLCopen XML document .....	51
Figure A.5 – Example of referencing an external document .....	52
Figure A.6 – XML text of the example .....	52
Figure A.7 – Example of referencing a CAEX attribute to an item in an external document.....	53
Figure A.8 – XML text of the example .....	53
Figure A.9 – Relations in AML .....	54
Figure A.10 – XML description of the relations example .....	55
Figure A.11 – XML text of the SystemUnitClassLib of the relations example .....	55
Figure A.12 – XML text of the InstanceHierarchy of the relations example .....	56
Figure A.2.1 – Port concept .....	57
Figure A.2.2 – Example describing the AML Port concept .....	58
Figure A.2.3 – XML description of the AML Port concept.....	59
Figure A.2.4 – XML text describing the AML Port concept.....	60
Figure A.2.5 – Definition of a user-defined AML Port class “UserDefinedPort”.....	60
Figure A.2.6 – AML Facet example .....	61
Figure A.2.7 – XML text of the AML Facet example .....	62

Figure A.2.8 – AML Group example .....	63
Figure A.2.9 – XML text for the AML Group example.....	63
Figure A.2.10 – Combination of the Facet and Group concept .....	64
Figure A.2.11 – XML text view for the combined Facet-Group example.....	65
Figure A.2.12 – Generic HMI template “B” visualizing a process variable “Y” of a conveyor.....	65
Figure A.2.13 – Generated HMI result “B” visualizing both conveyors with individual process variables.....	66
Figure A.2.14 – Base elements of the Product-Process-Resource concept .....	67
Figure A.2.15 – PPRConnector interface .....	67
Figure A.2.16 – Example for the Product-Process-Resource concept.....	68
Figure A.2.17 – AML roles required for the Process-Product-Resource concept .....	68
Figure A.2.18 – Elements of the example.....	69
Figure A.2.19 – Links within the example .....	69
Figure A.2.20 – Links of the resource centric view on the example .....	70
Figure A.2.21 – InstanceHierarchy of the example in AML .....	71
Figure A.2.22 – InternalElements of the example .....	72
Figure A.2.23 – InternalLinks of the example .....	72
Figure A.2.24 – InstanceHierarchy of the example in XML .....	73
Figure A.2.25 – Example describing the AML multilingual expression concept .....	74
Figure A.2.26 – XML description of the AML multilingual expression concept.....	74
Figure A.2.27 – XML text describing the AML multilingual expression concept.....	75
Figure A.2.28 – AML model of a multilingual AttributeType .....	75
Figure A.2.29 – XML code of the a multilingual AttributeType .....	75
Figure A.2.30 – Attribute list “SupportedFrequencies” .....	76
Figure A.2.31 – XML code for the attribute list “SupportedFrequencies” .....	76
Figure A.2.32 – Example CAEX model of the array “Edges”.....	77
Figure A.2.33 - XML code for the attribute array “Edges” .....	78
Figure B.1 - InstanceHierarchy of the ecample in XML .....	79

**List of tables**

Table 1 - Abbreviations .....	13
Table 2 – Interface classes of the AutomationMLInterfaceClassLib .....	21
Table 3 – InterfaceClass AutomationMLBaseInterface .....	23
Table 4 – InterfaceClass Order .....	23
Table 5 – Optional attributes for AML Port interfaces.....	24
Table 6 – InterfaceClass PPRConnector .....	24
Table 7 – InterfaceClass ExternalDataConnector .....	25
Table 8 – InterfaceClass COLLADAInterface .....	25
Table 9 – InterfaceClass PLCOpenXMLInterface .....	26
Table 10 – InterfaceClass ExternalDataReference .....	26
Table 11 – InterfaceClass Communication .....	26
Table 12 – InterfaceClass SignalInterface .....	27
Table 13 – RoleClass AutomationMLBaseRole.....	29
Table 14 – RoleClass Group.....	30
Table 15 – RoleClass Facet.....	30
Table 16 – RoleClass Resource .....	31
Table 17 – RoleClass Product .....	31
Table 18 – RoleClass Process .....	31
Table 19 – RoleClass Structure .....	32
Table 20 – RoleClass ProductStructure .....	32
Table 21 – RoleClass ProcessStructure .....	33
Table 22 – RoleClass ResourceStructure .....	33
Table 23 – RoleClass ExternalData.....	33
Table 24 – Attribute Types of the AutomationMLBaseAttributeTypeLib.....	34
Table 25 – Sub-attributes of the attribute “Cardinality” .....	36



## 1 Introduction

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

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

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

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

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

The core of AML is the top-level data format CAEX. CAEX is utilized to interconnect the different data formats. Therefore, AML has inherent distributed document architecture.

Figure 1.1 illustrates the basic AML architecture and the distribution of topology, geometry, kinematics, and logic information.

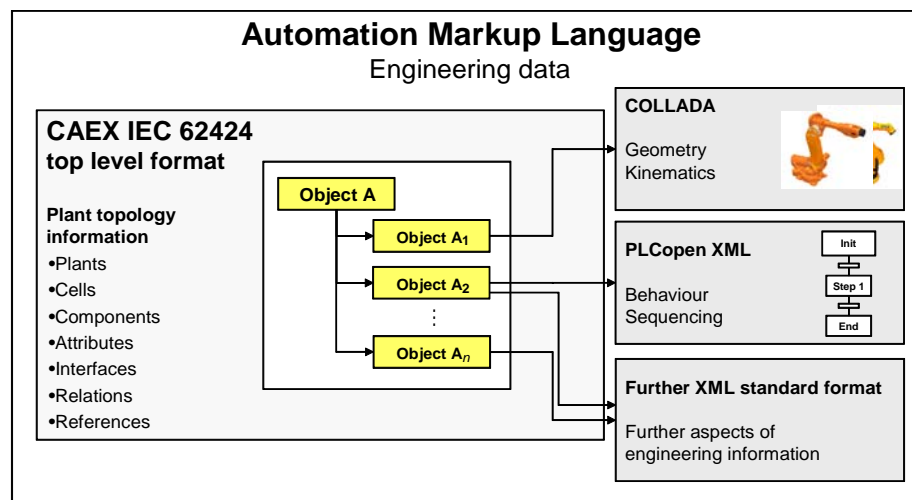


Figure 1.1 – Overview of the engineering data exchange format AML

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

- IEC 62714-1: Architecture and general requirements

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

- IEC 62714-2: Role class libraries

This part specifies additional AML libraries.

- IEC 62714-3: Geometry and kinematics

This part specifies the modelling of geometry and kinematics information.

- IEC 62714-4: Logic

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

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

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

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

Annex B gives an informative XML representation of the libraries defined in this whitepaper.

## 2 Scope

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

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

## 3 Normative references

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

IEC 62424:2015, *Representation of process control engineering – Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools*

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

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

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

COLLADA 1.4.1: March 2008, COLLADA – Digital Asset Schema Release 1.4.1  
(available at <[http://www.khronos.org/files/collada\\_spec\\_1\\_4.pdf](http://www.khronos.org/files/collada_spec_1_4.pdf)>)

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

PLCopen XML 2.0: December 3rd 2008 and PLCopen XML 2.0.1: May 8th 2009, XML formats for IEC 61131-3

(available at <<http://www.plcopen.org/>>)

## 4 Terms, definitions, and abbreviations

### 4.1 Terms and definitions

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

#### 4.1.1 AML

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

#### 4.1.2 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.

#### 4.1.3 AML object

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

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

#### 4.1.4 AML class

predefined AML object 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

#### 4.1.5 AML attribute

property 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 62424:2015, A.2.4.

#### 4.1.6 AML document

certain CAEX document following IEC 62714 including all referenced sub documents

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

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

#### 4.1.7 AML file

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

#### 4.1.8 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.

#### 4.1.9 AML library

library containing AML classes

#### 4.1.10 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.

#### 4.1.11 AML Group

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

#### 4.1.12 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

#### 4.1.13 CAEX

neutral XML based data format

Note 1 to entry: CAEX is a neutral data format according to IEC 62424:2015, Clause 7, Annex A and Annex C

#### 4.1.14 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.

#### 4.1.15 Universal unique identifier

UUID

unique identifier for AML objects

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

#### 4.1.16 Global unique identifier

GUID

implementation of a UUID

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

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

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

#### 4.1.17 Inheritance relation

relation between two AML classes

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

#### 4.1.18 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.

#### 4.1.19 Topology

hierarchical structure of a system, visualizable as object tree

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

#### 4.1.20 Plant topology

hierarchical structure of a plant, visualizable as object tree

**4.1.21 Publish, verb**

to model a data structure of an external document for usage within CAEX

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

**4.1.22 Relation**

association between CAEX objects

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

**4.1.23 Link**

connection between objects of type CAEX ExternalInterface

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

**4.1.24 Reference**

association between a CAEX InternalElement and externally stored information

**4.2 Abbreviations**

*Table 1 - Abbreviations*

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

## 5 AML architecture specifications

### 5.1 General

The centre of AML is the top-level data format CAEX, a neutral data format according to IEC 62424:2015, Clause 7, Annex A and Annex C, that interconnects established data formats for the engineering aspects for topology, geometry, kinematics, behaviour and sequencing information. Therefore, a basic characteristic of AML is an inherent distributed document architecture focussing on the above-mentioned engineering aspects.

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

### 5.2 General AML architecture

Regarding the general AML architecture, the following provisions apply:

**Plant topology information:** The plant topology acts as the top-level data structure of the plant engineering information and shall be modelled by means of the data format CAEX according to IEC 62424:2015, 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 62424:2015, A2.8.7.

NOTE 1 According to IEC 62424:2015, A.2.14, an AML object with a relation to another AML object is called “mirror object” whereas the related AML object is called “master object”. The mirror object is considered to be identical to the master object. This enables placing one object instance into different plant hierarchies and thus allows modelling of complex object networks with crossed structures.

NOTE 2 IEC 62714 does not syntactically modify the CAEX data format. An informative overview and additional examples regarding the plant topology are provided in A.1.2 and in IEC 62424:2015, 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.1.5. References and publishing of interfaces are described in additional parts of IEC 62714.

NOTE 4 Relations depict associations between CAEX objects.

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

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

---

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

NOTE 7 Logic information describes sequences of actions and the internal behaviour of objects including I/O connections and logical variables. IEC 62714 does not modify the PLCopen XML format. An informative overview of how to reference logic information is provided in A.1.4. Details are 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 IEC62424:2016;
- 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 IEC62424:2016, A2.2.3,.
- The value of this element shall be "AutomationML 2.1" 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 5.1 illustrates the XML text for a CAEX document following the AML version 3.0.

```
<SourceDocumentInformation OriginID="IEC SC65E WG 9" OriginName="IEC
SC65E WG 9" OriginVersion="2.1.0" LastWritingDateTime="2015-10-06T14:43:00.0Z"
OriginProjectID="Automation Markup Language Standard Library" OriginRelease="
2.1.0" OriginVendor="IEC" OriginVendorURL="www.iec.ch" OriginProjectTitle="
Automation Markup Language Standard Libraries"/>
```

Figure 5.1 - 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 IEC62424:2016, 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 hinder the iterative data exchange with the other tools: hence the removal of writer information of other tools is not recommended.
- This standard recommends to use the optional CAEX element `SourceObjectInformation` with its attributes `OriginID` and `SourceObjID` according to IEC 62424:2015, A.2.2.7, in order to identify the source tool of each AML Object instance (`InternalElement`, `ExternalInterface`).

Figure 5.2 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 IEC62714.

```
<SourceDocumentInformation OriginID="IEC SC65E WG 9" OriginName="IEC SC65E WG 9"
OriginVersion="2.1.0" LastWritingDateTime="2015-10-06T14:43:00.0Z" OriginProjectID="Automation
Markup Language Standard Library" OriginRelease="2.1.0" OriginVendor="IEC"
OriginVendorURL="www.iec.ch" OriginProjectTitle="Automation Markup Language Standard Libraries"/>
```

Figure 5.2 - 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 whitepaper 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.1.5.

### 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 62424:2015, 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 define the semantic of the interface and thus the semantic of the link. A link between interfaces without a reference to an interface class has no semantics.

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

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 derivatives.

NOTE 3 The standard interface class “PLCopenXMLInterface” is specified in 6.3.8. Details are specified in IEC 62714-4.

Figure 5.3a) describes an example comprising a robot “Rob1” and a PLC “PLC1”, each with one signal interface that are connected. Figure 5.3b) depicts this example as an object hierarchy.

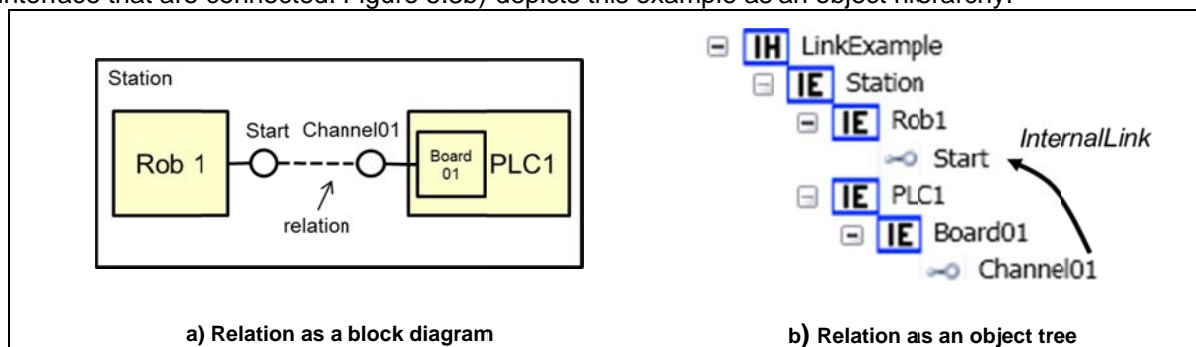


Figure 5.3 - Example of a relation as block diagram and as object tree

Figure 5.4 and Figure 5.5 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 are reduced with “/...” in order to increase the readability.

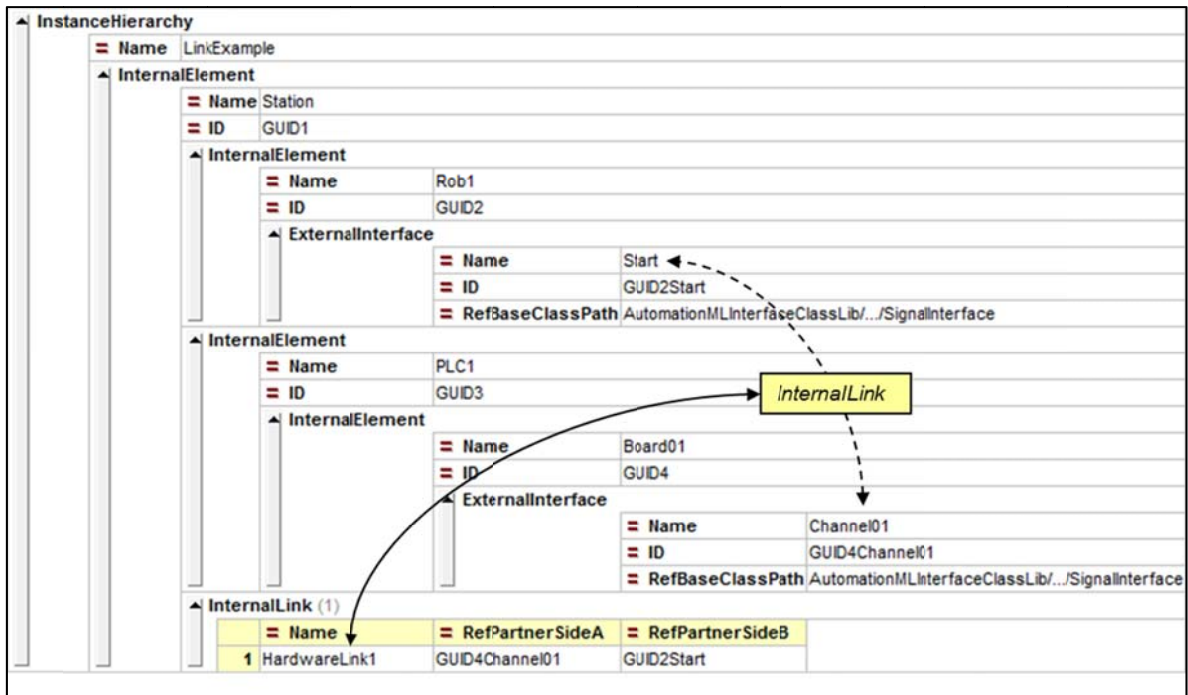


Figure 5.4 - Example relation between the objects "PLC1" and "Rob1"

```

<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>

```

Figure 5.5 - XML text of the example relation between the objects "PLC1" and "Rob1"

## 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 derivatives. 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 “PLCopen-XMLInterface” or one of its derivatives. 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 IEC62714

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 this standard. For these extensions, the following provisions apply:

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

The standard interface class “ExternalDataConnector” shall only be used for document types included in IEC 62714.

### 5.6.5 Referencing documents outside of the scope of IEC62714

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:

- Each external document 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: Each document can contain contents of several types, e.g. bill of material and user manual.

NOTE: Each document can reference to several files if necessary, e.g. if it is 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”.
- The 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”.
- The ExternalInterface shall model the type of document by means of the predefined CAEX attribute “MIMEType” of type “MIMEType”.

Additional information and an example is 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 related items in external documents (e.g. a variable in a PLCOpen XML document, an item in a COLLADA 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 5.6.2-5.6.5.
- This ExternalInterface is valid for one item only. If multiple items shall be modelled, each item shall be referenced by means of another ExternalInterface.
- Beside the attribute refURI, which references the item within the external document, the external interface shall contain a second CAEX attribute of type "AssociatedValue" which has one nested attribute. The nested attribute shall mirror the desired CAEX attribute. Only one nested attribute is allowed. Associating multiple attributes shall modelled via multiple interfaces.

Additional information and an example is 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”.
- “AutomationMLBaseInterface”.

### 6.3 AML interface class library – AutomationMLInterfaceClassLib

#### 6.3.1 General

The following “AutomationMLInterfaceClassLib” is modelled according to IEC 62424:2015, Clause 7, Annex A and Annex C. IEC 62714 utilizes the CAEX interface concept. User-defined extensions of this AML library are allowed as specified in 7.4.

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
<pre> classDiagram     class AutomationMLInterfaceClassLib {         AutomationMLBaseInterface     }     class AutomationMLBaseInterface {         Order         Port         PPRConnector         ExternalDataConnector         COLLADAInterface         PLCopenXMLInterface         ExternalDataReference         Communication         SignalInterface     }     AutomationMLInterfaceClassLib --&gt; AutomationMLBaseInterface   </pre>	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 6.1 shows a table view and Figure 6.2 the XML text of the standard AML InterfaceClassLib. Subclauses 6.3.2 to 6.3.10 provide detail information about the classes.

InterfaceClassLib

Name	AutomationMLInterfaceClassLib		
Description	Standard Automation Markup Language Interface Class Library		
Version	2.2.5		
InterfaceClass			
AutomationMLBaseInterface			
InterfaceClass			
Name	Order		
RefBaseClassPath	AutomationMLBaseInterface		
Attribute (1)			
1	Direction	xs:string	AutomationMLBaseAttributeTypeLib/Direction
InterfaceClass			
Name	Port		
RefBaseClassPath	AutomationMLBaseInterface		
Attribute			
Name	Direction		
AttributeDataType	xs:string		
RefAttributeType	AutomationMLBaseAttributeTypeLib/Direction		
Constraint			
Name	AllowedValues		
NominalScaledType			
RequiredValue (3)			
1	In		
2	Out		
3	InOut		
Attribute			
Name	Cardinality		
RefAttributeType	AutomationMLBaseAttributeTypeLib/Cardinality		
Attribute (2)			
1	MinOccur	xs:unsignedInt	
2	MaxOccur	xs:unsignedInt	
Attribute			
Name	Category		
AttributeDataType	xs:string		
RefAttributeType	AutomationMLBaseAttributeTypeLib/Category		
InterfaceClass			
Name	PPRConnector		
RefBaseClassPath	AutomationMLBaseInterface		
InterfaceClass			
Name	ExternalDataConnector		
RefBaseClassPath	AutomationMLBaseInterface		
Attribute (1)			
1	refURI	xs:anyURI	AutomationMLBaseAttributeTypeLib/refURI
InterfaceClass			
Name	COLLADAInterface		
RefBaseClassPath	ExternalDataConnector		
InterfaceClass			
Name	PLCopenXMLInterface		
RefBaseClassPath	ExternalDataConnector		
InterfaceClass			
Name	ExternalDataReference		
RefBaseClassPath	ExternalDataConnector		
Attribute			
Name	MIMType		
AttributeDataType	xs:string		
RefAttributeType	AutomationMLBaseAttributeTypeLib/MIMType		
InterfaceClass			
Name	Communication		
RefBaseClassPath	AutomationMLBaseInterface		
InterfaceClass (1)			
1	SignalInterface	Communication	

Figure 6.1 – AML basic interface class library



```

<InterfaceClassLib Name="AutomationMLInterfaceClassLib">
  <Description>Standard Automation Markup Language Interface Class Library</Description>
  <Version>2.2.5</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>

```

Figure 6.2 – 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

<b>Class name</b>	Order
<b>Description</b>	The interface class “Order” is an abstract class that shall be used for the description of orders, e.g. a successor or a predecessor.
<b>Parent class</b>	AutomationMLInterfaceClassLib/AutomationMLBaseInterface
<b>Path for element reference</b>	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Order
<b>Attributes</b>	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*

<b>Class name</b>	Port
<b>Description</b>	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.
<b>Parent class</b>	AutomationMLInterfaceClassLib/AutomationMLBaseInterface
<b>Path for element reference</b>	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Port
<b>Attributes</b>	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*



<b>Class name</b>	PPRConnector	
<b>Description</b>	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.	
<b>Parent class</b>	AutomationMLInterfaceClassLib/AutomationMLBaseInterface	
<b>Path for element reference</b>	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/PPRInterface	
<b>Attributes</b>	None	

### 6.3.6 InterfaceClass ExternalDataConnector

Table 7 specifies the interface class “ExternalDataConnector”.

Table 7 – InterfaceClass ExternalDataConnector

<b>Class name</b>	ExternalDataConnector	
<b>Description</b>	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.	
<b>Parent class</b>	AutomationMLInterfaceClassLib/AutomationMLBaseInterface	
<b>Path for element reference</b>	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector	
<b>Attributes</b>	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

<b>Class name</b>	COLLADAInterface	
<b>Description</b>	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.	
<b>Parent class</b>	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector	
<b>Path for element reference</b>	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/COLLADAInterface	
<b>Attributes</b>	None	

### 6.3.8 InterfaceClass PLCopenXMLInterface

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

Table 9 – InterfaceClass PLCopenXMLInterface

<b>Class name</b>	PLCopenXMLInterface	
<b>Description</b>	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.	
<b>Parent class</b>	AutomationMLBaseInterface/ExternalDataConnector	
<b>Path for element reference</b>	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/PLCopenXMLInterface	
<b>Attributes</b>	None	

### 6.3.9 InterfaceClass ExternalDataReference

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

Table 10 – InterfaceClass ExternalDataReference

<b>Class name</b>	ExternalDataReference	
<b>Description</b>	The interface class “ExternalDataReference” shall be used in order to reference external documents out of the scope of AutomationML. Details are specified in 5.6.5.	
<b>Parent class</b>	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector	
<b>Path for element reference</b>	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataReference	
<b>Attributes</b>	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

<b>Class name</b>	Communication	
<b>Description</b>	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.	
<b>Parent class</b>	AutomationMLInterfaceClassLib/AutomationMLBaseInterface	
<b>Path for element reference</b>	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication	
<b>Attributes</b>	None	

### 6.3.11 InterfaceClass SignalInterface

Table 12 specifies the interface class “SignalInterface”.

Table 12 – InterfaceClass SignalInterface

<b>Class name</b>	SignalInterface	
<b>Description</b>	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 5.3.	
<b>Parent class</b>	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication	
<b>Path for element reference</b>	AutomationMLInterfaceClassLib/AutomationMLBaseInterface//SignalInterface	
<b>Attributes</b>	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 6.3 to Figure 6.5 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.

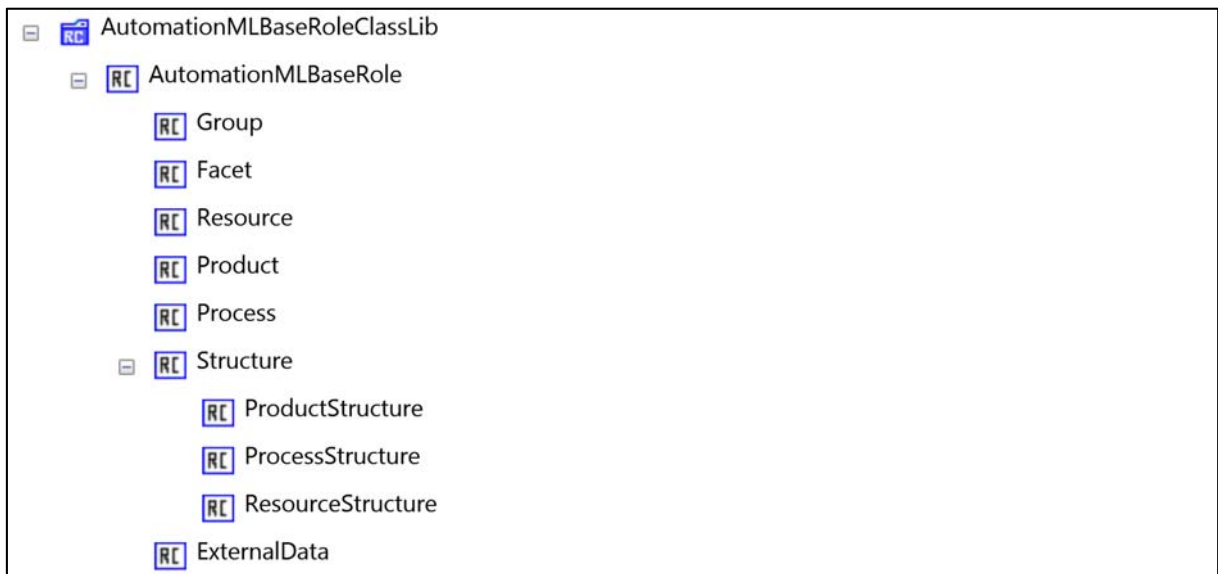


Figure 6.3 - AML basic role class library

RoleClassLib	
Name	AutomationMLBaseRoleClassLib
Description	Automation Markup Language base role class library
Version	2.2.5
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)	
1	ProductStructure
2	ProcessStructure
3	ResourceStructure
RoleClass	
Name	ExternalData
RefBaseClassPath	AutomationMLBaseRole

Figure 6.4 - AutomationMLBaseRoleClassLib

```

<RoleClassLib Name="AutomationMLBaseRoleClassLib">
  <Description>Automation Markup Language base role class library</Description>
  <Version>2.2.5</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>

```

Figure 6.5 – XML text of the AutomationMLBaseRoleClassLib

## 6.4.2 RoleClass AutomationMLBaseRole

Table 13 specifies the role class “AutomationMLBaseRole”.

Table 13 – RoleClass AutomationMLBaseRole

<b>Class name</b>	AutomationMLBaseRole	
<b>Description</b>	The role class “AutomationMLBaseRole” is a basic abstract role type and the base class for all standard or user-defined role classes.	
<b>Parent class</b>	None	
<b>Path for element reference</b>	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
<b>Attributes</b>	None	

#### 6.4.3 RoleClass Group

Table 14 specifies the role class “Group”.

Table 14 – RoleClass Group

<b>Class name</b>	Group	
<b>Description</b>	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.	
<b>Parent class</b>	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
<b>Path for element reference</b>	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Group	
<b>Attributes</b>	<b>Name:</b> AssociatedFacet <b>RefAttributeType:</b> AutomationMLBaseAttributeTypeLib/AssociatedFacet Semantics: see 6.5.2	

#### 6.4.4 RoleClass Facet

Table 15 specifies the role class “Facet”.

Table 15 – RoleClass Facet

<b>Class name</b>	Facet	
<b>Description</b>	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.	
<b>Parent class</b>	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
<b>Path for element reference</b>	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Facet	
<b>Attributes</b>	None	

#### 6.4.5 RoleClass Resource

Table 16 specifies the role class “Resource”.

Table 16 – RoleClass Resource

<b>Class name</b>	Resource	
<b>Description</b>	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.	
<b>Parent class</b>	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
<b>Path for element reference</b>	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource	
<b>Attributes</b>	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

<b>Class name</b>	Product	
<b>Description</b>	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.	
<b>Parent class</b>	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
<b>Path for element reference</b>	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Product	
<b>Attributes</b>	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

<b>Class name</b>	Process	
<b>Description</b>	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.	
<b>Parent class</b>	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
<b>Path for element reference</b>	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Process	
<b>Attributes</b>	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

<b>Class name</b>	Structure	
<b>Description</b>	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.	
<b>Parent class</b>	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
<b>Path for element reference</b>	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure	
<b>Attributes</b>	None	

#### 6.4.9 RoleClass ProductStructure

Table 20 specifies the role class “ProductStructure”.

Table 20 – RoleClass ProductStructure

<b>Class name</b>	ProductStructure	
<b>Description</b>	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.	
<b>Parent class</b>	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure	
<b>Path for element reference</b>	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure/ProductStructure	
<b>Attributes</b>	None	



#### 6.4.10 RoleClass ProcessStructure

Table 21 specifies the role class “ProcessStructure”.

Table 21 – RoleClass ProcessStructure

<b>Class name</b>	ProcessStructure	
<b>Description</b>	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.	
<b>Parent class</b>	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure	
<b>Path for element reference</b>	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure /ProcessStructure	
<b>Attributes</b>	None	

#### 6.4.11 RoleClass ResourceStructure

Table 22 specifies the role class “ResourceStructure”.

Table 22 – RoleClass ResourceStructure

<b>Class name</b>	ResourceStructure	
<b>Description</b>	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.	
<b>Parent class</b>	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure	
<b>Path for element reference</b>	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure/ResourceStructure	
<b>Attributes</b>	None	

#### 6.4.12 RoleClass ExternalData

Table 23 specifies the role class “ExternalData”.

Table 23 – RoleClass ExternalData

<b>Class name</b>	ExternalData	
<b>Description</b>	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.	
<b>Parent class</b>	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
<b>Path for element reference</b>	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/ExternalData	
<b>Attributes</b>	None	

## 6.5 AML basic attribute type library

### 6.5.1 General

This subclause defines AML basic attribute types, required by the standard AML classes defined in Subclause 6.3 and 6.4.

### 6.5.2 AML basic attribute type library AutomationMLBaseAttributeTypeLib

Table 24 specifies the attribute types of the AutomationMLBaseAttributeTypeLib.

*Table 24 – Attribute Types of the AutomationMLBaseAttributeTypeLib*

Attribute Name	Semantics
Direction	<p>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”.</p> <p>CAEX Interfaces using this attribute follow the following provisions:</p> <ul style="list-style-type: none"> <li>• Interfaces with the direction “In” shall only be connected to interfaces with the direction “Out” or “InOut”.</li> <li>• Interfaces with the direction “Out” shall only be connected to interfaces with the direction “In” or “InOut”.</li> </ul> <p>This information can be used e.g. in order to prove the validity of a connection.</p> <p>Examples:</p> <ul style="list-style-type: none"> <li>• Direction = “Out” (e.g. a plug)</li> <li>• Direction = “In” (e.g. a socket)</li> <li>• Direction = “InOut”</li> </ul> <p>NOTE The validity of those connections is outside the scope of IEC 62714, but is a tool functionality.</p> <p><b>AttributeDataType:</b> xs:string  <b>Path:</b> AutomationMLBaseAttributeTypeLib/Direction</p>
Cardinality	<p>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.</p> <p>The attribute Cardinality itself is a complex attribute and shall not have a value. The corresponding sub-attributes are described in Table 25.</p> <p><b>AttributeDataType:</b> This attribute has no AttributeDataType since the attribute has no value.</p> <p><b>Path:</b> AutomationMLBaseAttributeTypeLib/Category</p>
Category	<p>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.</p> <p>Example: Category = “MaterialFlow”.</p> <p><b>AttributeDataType:</b> xs:string  <b>Path:</b> AutomationMLBaseAttributeTypeLib/Category</p>
refURI	<p>This attribute shall be used in order to store a path to an external document.</p> <p><b>AttributeDataType:</b> xs:anyURI  <b>Path:</b> AutomationMLBaseAttributeTypeLib/refURI</p>
AssociatedFacet	<p>The attribute “AssociatedFacet” shall be used for the definition of the name of a related Facet. The Facet concept is described in 8.3.</p> <p><b>Example:</b> AssociatedFacet = “PLCFacet”.</p> <p><b>AttributeDataType:</b> xs:string  <b>Path:</b> AutomationMLBaseAttributeTypeLib/AssociatedFacet</p>
ListType	<p>The attribute “ListType” shall be used for attributes that contain an unsorted list of attributes. The concept is described in A.2.7.</p> <p><b>AttributeDataType:</b> empty  <b>Path:</b> AutomationMLBaseAttributeTypeLib/ListType</p>
OrderedListType	<p>The attribute “OrderedListType” shall be used for attributes that contain a sorted list of attributes. The concept is described in A.2.7.</p> <p><b>AttributeDataType:</b> empty  <b>Path:</b> AutomationMLBaseAttributeTypeLib/OrderedListType</p>
LocalizedAttribute	<p>The attribute “LocalizedAttribute” shall be used for sub-attributes describing a language alternative of its parent attribute. The language according to RFC5654 shall be used as name of the attribute. The concept is described in A.2.6.</p> <p><b>AttributeDataType:</b> xs:string  <b>Path:</b> AutomationMLBaseAttributeTypeLib/LocalizedAttribute</p>

AssociatedValue	<p>The AssociatedValue contains a mirror attribute, which allows to interconnect a CAEX attribute to another CAEX object; e.g. an interface with a reference to an attribute within an external document. The concept is described in A.1.6.</p> <p><b>AttributeDataType:</b> empty</p> <p><b>Parent:</b> AutomationMLBaseAttributeTypeLib</p> <p><b>Path:</b> AutomationMLBaseAttributeTypeLib/AssociatedValue</p>
MIMETYPE	<p>The MIMETYPE describes the MIMETYPE of a referenced document.</p> <p>The attribute shall have values according to the Multipurpose Internet Mail Extensions (MIME) standard.</p> <p><b>Examples:</b></p> <p>MIMETYPE = "application/pdf" - this means that this document is of file type pdf.</p> <p>MIMETYPE = "application/xml" – this means that this document of type xml.</p> <p><b>AttributeDataType:</b> xs:string</p> <p><b>Parent:</b> AutomationMLBaseAttributeTypeLib</p> <p><b>Path:</b> AutomationMLBaseAttributeTypeLib/MIMETYPE</p>
DocLang	<p>The DocLang describes the language of a referenced document.</p> <p>The attribute shall have a value according to the RFC1766 standard.</p> <p><b>Example:</b> DocLang = fr-FR - This means that this document is in French language valid in France.</p> <p><b>AttributeDataType:</b> xs:string</p> <p><b>Parent:</b> AutomationMLBaseAttributeTypeLib</p> <p><b>Path:</b> AutomationMLBaseAttributeTypeLib/DocLang</p>

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 MinOccur, or 0 which means infinite.	MaxOccur = 3 This means that this Port can only be connected with a maximum of three other ports.

AttributeTypeLib	
Name	AutomationMLBaseAttributeTypeLib
Description	Standard Automation Markup Language Attribute Type Library
Version	1.0
AttributeType	
Name	Direction
AttributeDataType	xs:string
Constraint	
AllowedValues	
NominalScaledType	
RequiredValue (3)	
	1 In
	2 Out
	3 InOut
AttributeType	
Name	Cardinality
Attribute (2)	
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	AssociatedValue
AttributeType	
Name	MIMEType
AttributeDataType	xs:string
AttributeType	
Name	DocLang
AttributeDataType	xs:string

Figure 6.6 – AML basic attribute type library

```
<AttributeTypeLib Name="AutomationMLBaseAttributeTypeLib">
  <Description>Standard Automation Markup Language Attribute Type Library</Description>
  <Version>1.0</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="AssociatedValue"/>
  <AttributeType Name="MIMEType" AttributeDataType="xs:string"/>
  <AttributeType Name="DocLang" AttributeDataType="xs:string"/>
</AttributeTypeLib>
```

Figure 6.7 – XML text of the AutomationMLBaseAttributeTypeLib

## 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. The AML top-level data format CAEX provides mechanisms for modelling of user-defined data.

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

AML allows defining a relation between user-defined data and standard data by means of the Role Concept, the 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 are AML standard attributes. Attributes which are not defined in IEC 62714 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 62424:2015, A.2.4.
- If units are required, user-defined attributes shall base on the same unit system. This whitepaper 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 7.1 gives an example of a user-defined object “Object01” with a user-defined attribute “Length”.

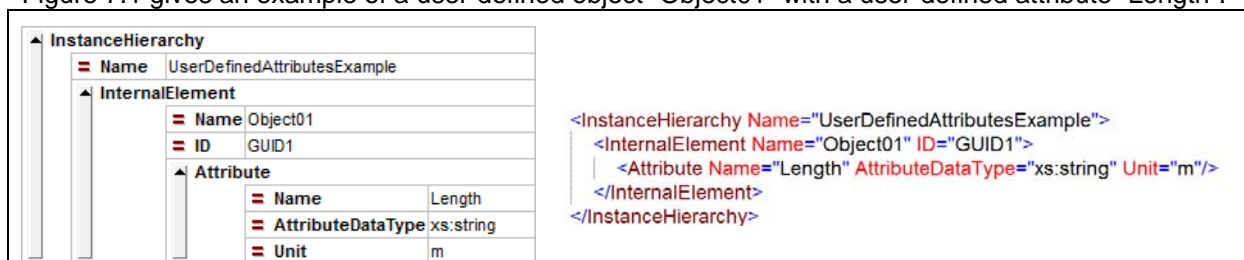


Figure 7.1 – Example of a user-defined attribute

### 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 62424:2015, 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 today's or future attribute standards. This is the basis for the storage of agreed attribute semantics and enables independence from language and naming conventions.

**Examples:** Figure 7.2 and Figure 7.3 illustrates 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.

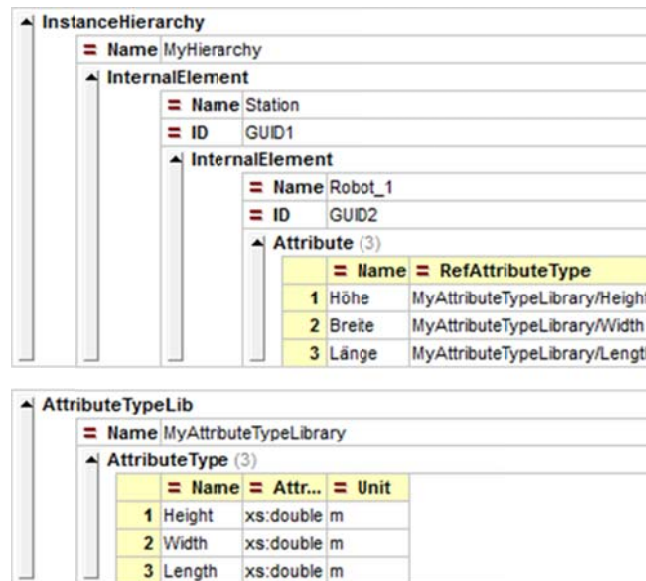


Figure 7.2 – Examples for user-defined AttributeTypes

```
<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>
</InstanceHierarchy>

<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>
```

Figure 7.3 – XML code of the examples for user-defined AttributeTypes

## 7.4 User-defined InterfaceClasses

All InterfaceClasses defined in IEC 62714 are standard AML interface classes. All InterfaceClasses not defined in IEC 62714 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 62424:2015, 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 7.4 and Figure 7.5 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.



▲ InterfaceClassLib			
= Name		UserDefinedClassLib	
▲ InterfaceClass			
= Name		MyDigitalInput	
= RefBaseClassPath		AutomationMLInterfaceClassLib/.../SignalInterface	
() Version		1.0	
▲ Attribute (3)			
	= Name	= AttributeDataType	() Value
1	Type	xs:string	Digital
2	Direction	xs:string	In
3	Enabled	xs:boolean	true

Figure 7.4 – 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>
</InterfaceClassLib>

```

Figure 7.5 – 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 are standard AML role classes. All role classes not defined in IEC 62714 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 62424:2015, A.2.6.

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

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

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

Figure 7.6 and Figure 7.7 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.

RoleClassLib		
Name	UserDefinedRoleClassLib	
Version	1.0	
RoleClass (1)		
	Name	RefBaseClassPath
1	Fence	AutomationMLRoleClassLib/AutomationMLBaseRole/Resource

Figure 7.6 – 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>
```

Figure 7.7 – 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 user-defined.

Regarding user-defined SystemUnitClasses, the following provisions apply:

- User-defined SystemUnitClasses shall be stored in AML according to the CAEX SystemUnitClass definition in IEC 62424:2015, 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.

Note 1: It is recommended to assign user-defined SystemUnitClasses to a role class. This serves the automatic interpretability.

**Examples:** Figure 7.8 and Figure 7.9 illustrates the definition of a user-defined SystemUnitClass by means of two different examples.

- The SystemUnitClass "Robot1234" depicts a user-defined class which supports the role "Resource" of the AML standard RoleClassLib. This class can therefore 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.

SystemUnitClassLib	
Name	UserDefinedSystemUnitClassLib
SystemUnitClass	
Name	Robot1234
ID	GUID1
Version	1.0
SupportedRoleClass	
RefRoleClassPath	AutomationMLRoleClassLib/AutomationMLBaseRole/Resource
SystemUnitClass	
Name	SpecialRobot1234
RefBaseClassPath	UserDefinedSystemUnitClassLib/Robot1234

Figure 7.8 – 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>
```

Figure 7.9 – 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 whitepaper does not restrict the depth of the hierarchy levels.
- This whitepaper does not restrict the architecture rules of a hierarchy.
- This whitepaper 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 7.10 depicts an example project hierarchy that comprises a line “L001” with a station “S001” containing two robots “R0010\_D” and “R0020\_D” as well as a conveyor “RF010” and a PLC “P001”.

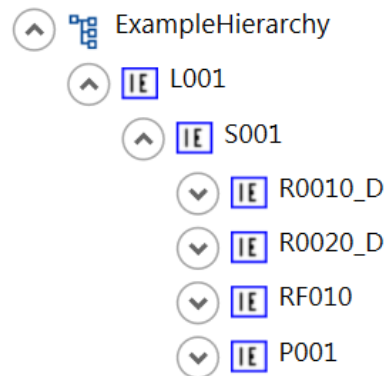


Figure 7.10 – Example of a user-defined InstanceHierarchy

Figure 7.11 shows the AML representation of this structure. According to IEC 62424:2015, 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>
```

Figure 7.11 – AML representation of a user-defined InstanceHierarchy

## 8 Extended AML concepts

### 8.1 General overview

This whitepaper defines extended concepts for the modelling of specific engineering aspects. An informative overview and examples are provided in 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 whitepaper 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 62424:2015, 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 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 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 62424:2015, 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 including examples 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 RFC5654, 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 62424:2015, 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 referring to the AttributeType “ListType”.
- If the list is ordered, this list attribute shall referring 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 referring to the AttributeType “ListType” or “OrderedListType”.

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, the present standard supports the Open Packaging Conventions (OPC) as container format. Following the OPC definitions, the following provision apply:

- AML container shall be stored as ECMA OPC archive which provides data compression.
- 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 standard 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 RoleClassLib’s, InterfaceClassLib’s, SystemUnitClassLib’s and/or AttributeTypeLibs. 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 the Mime specification .

**CAEX Scheme**

*Relationshiptype:* <http://schemas.automationml.org/container/relationship/CAEXScheme>

*Mime type:* "text/xml"



## A 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.

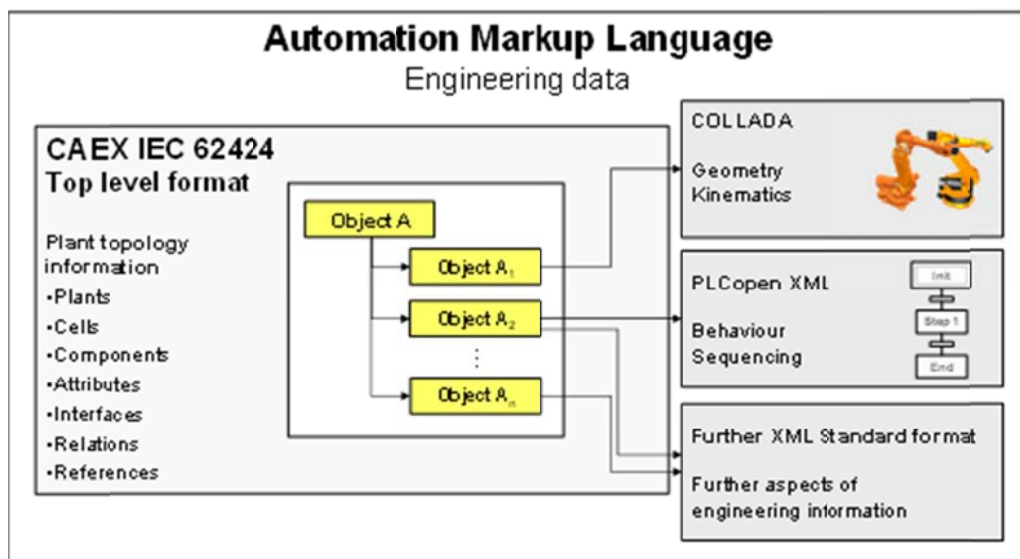


Figure A.1 - AML general architecture

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 62424:2015, 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.1.5 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 whitepaper.

### 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 4.1.19).

In order to store hierarchical plant structures, AML utilizes concepts provided by the top-level data format CAEX according to IEC 62424:2015, 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.

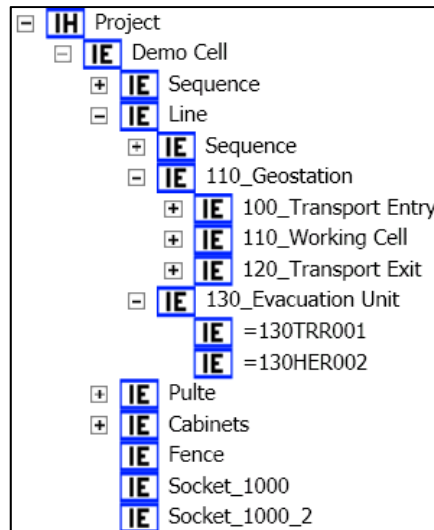


Figure A.2 - Plant topology with AML

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 user-defined 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 62424:2015 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 COL-LADA 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.

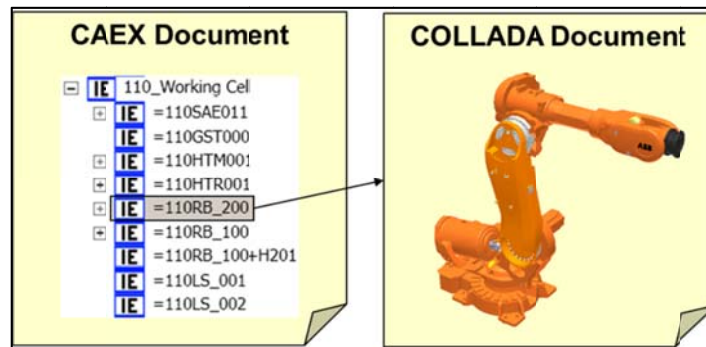


Figure A.3 - Reference from CAEX to a COLLADA document

The reference is modelled by means of a CAEX interface derived from the standard AML in-terface 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.

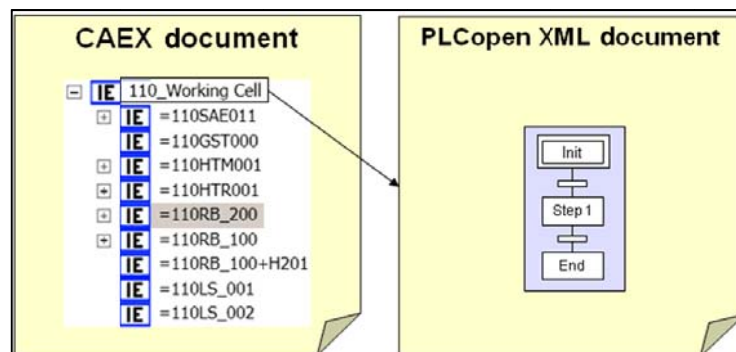


Figure A.4 – Reference from a CAEX to a PLCopen XML document

### A.1.5 Referencing documents outside of the scope of IEC62714

This standard defines a mechanism which allows to reference external documents which are not in the scope of this standard (e.g. documents, pdf files, Excel sheets, XML files etc.). Figure A.5 and Figure A.6 illustrate how to model this. For the modelling of this referencing, the InternalElement “TemperatureSensor” has an ExternalInterface of type ExternalDataReference which references the

file “example.xml”. The attribute “MIMEType” defines the document type, in this case it is of the “MIMEType application/xml”. Normative provisions are defined in 5.6.5.

▲ InstanceHierarchy			
= Name		Example ExternalDataReference	
▲ InternalElement			
= Name		TemperatureSensor	
= ID		GUID1	
▲ Attribute (2)			
	= Name	= Unit	⚙ Value
1	Temperature	°C	10
2	Color		Blue
▲ ExternalInterface			
= Name		ExternalItem	
= ID		GUID2	
= RefBaseClassPath		AutomationMLBaseInterfaceClassLib/.../ExternalDataReference	
▲ Attribute (2)			
	= Name	= RefAttributeType	⚙ Value
1	refURI	AutomationMLBaseAttributeTypeLib/refURI	./example.xml
2	MIMETYPE	AutomationMLBaseAttributeTypeLib/MIMETYPE	application/xml
▲ RoleRequirements			
= RefBaseRoleClassPath		AutomationMLBaseRoleClassLib/.../Resource	

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>
    <ExternalInterface Name="ExternalItem" ID="GUID2" 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/.../Resource"/>
  </InternalElement>
</InstanceHierarchy>

```

Figure A.6 – XML text of the example

#### A.1.6 Interlinking CAEX attributes and attributes in external documents

This standard defines a mechanism to associate the value of a CAEX Attribute to an item within an external document. Figure A.7 and Figure A.8 illustrate how to model this. This example is an extension of the example shown in subclause 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 subclause A.1.5 Figure A.5., for the modelling of this referencing, the InternalElement *ControlProgram* has an additional ExternalInterface of type *ExternalDataConnector* which references the item within the external document. Furthermore, this ExternalInterface has a second CAEX Attribute of CAEX AttributeType *AssociatedValue*. This just mirrors the CAEX attribute *Temperature*. This standard does not define the direction of this association, this depends on the use case.

Normative provisions are defined in 5.6.6

InstanceHierarchy			
= Name		ExampleAssociatedValue	
InternalElement			
= Name		TemperatureSensor	
= ID		GUID1	
Attribute (2)			
	= Name	= Unit	Value
1	Temperature	°C	10
2	Color		Blue
ExternalInterface			
= Name		ExternalItem	
= ID		GUID2	
= RefBaseClassPath		AutomationMLBaseInterfaceClassLib/.../ExternalDataReference	
Attribute (2)			
	= Name	= RefAttributeType	Value
1	refURI	AutomationMLBaseAttributeTypeLib/refURI	./example.xml
2	MIMETYPE	AutomationMLBaseAttributeTypeLib/MIMETYPE	application/xml
ExternalInterface			
= Name		ExternalItem	
= ID		GUID2	
= RefBaseClassPath		AutomationMLBaseInterfaceClassLib/.../ExternalDataReference	
Attribute			
	= Name	refURI	
	= RefAttributeType	AutomationMLBaseAttributeTypeLib/refURI	
	Value	./example.xml#referencedItem	
Attribute			
	= Name	AssociatedValue	
	= RefAttributeType	AutomationMLBaseAttributeTypeLib/AssociatedValue	
Attribute			
	= Name	Temperature	
	= RefAttributeType	GUID1/Temperatur	
RoleRequirements			
= RefBaseRoleClassPath		AutomationMLBaseRoleClassLib/.../Resource	

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

```

<InstanceHierarchy Name="ExampleAssociatedValue">
  <InternalElement Name="TemperatureSensor" ID="GUID1">
    <Attribute Name="Temperature" Unit="°C">
      <Value>10</Value>
    </Attribute>
    <Attribute Name="Color">
      <Value>Blue</Value>
    </Attribute>
    <ExternalInterface Name="ExternalItem" ID="GUID2" 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>
    <ExternalInterface Name="ExternalItem" ID="GUID2" RefBaseClassPath="AutomationMLBaseInterfaceClassLib/.../ExternalDataReference">
      <Attribute Name="refURI" RefAttributeType="AutomationMLBaseAttributeTypeLib/refURI">
        <Value>./example.xml#referencedItem</Value>
      </Attribute>
      <Attribute Name="AssociatedValue" RefAttributeType="AutomationMLBaseAttributeTypeLib/AssociatedValue">
        <Attribute Name="Temperature" RefAttributeType="GUID1/Temperatur"/>
      </Attribute>
    </ExternalInterface>
    <RoleRequirements RefBaseRoleClassPath="AutomationMLBaseRoleClassLib/.../Resource"/>
  </InternalElement>
</InstanceHierarchy>

```

Figure A.8 – XML text of the example

### 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 62424:2015, 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.

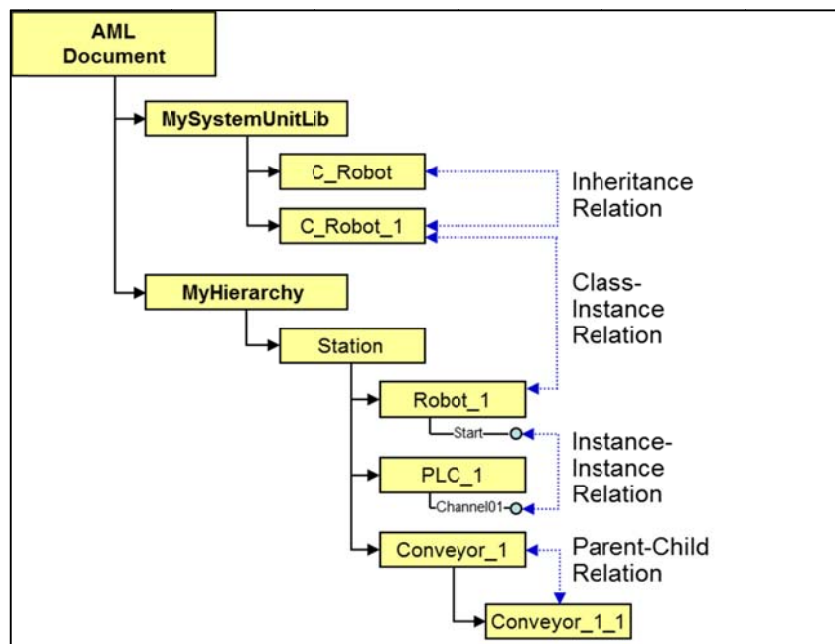


Figure A.9 – Relations in AML

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.

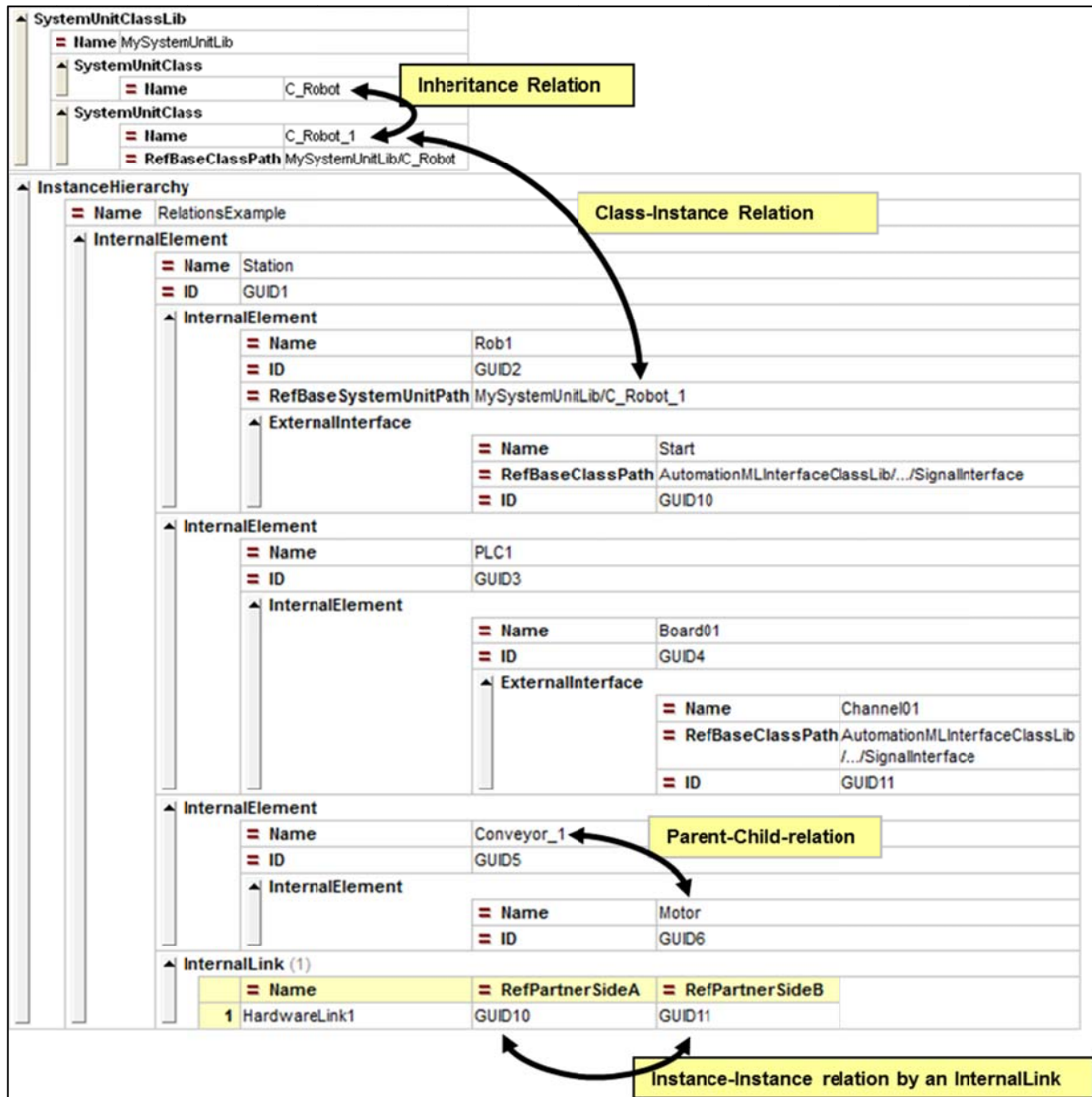


Figure A.10 – XML description of the relations example

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

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>
```

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.2.1). 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.

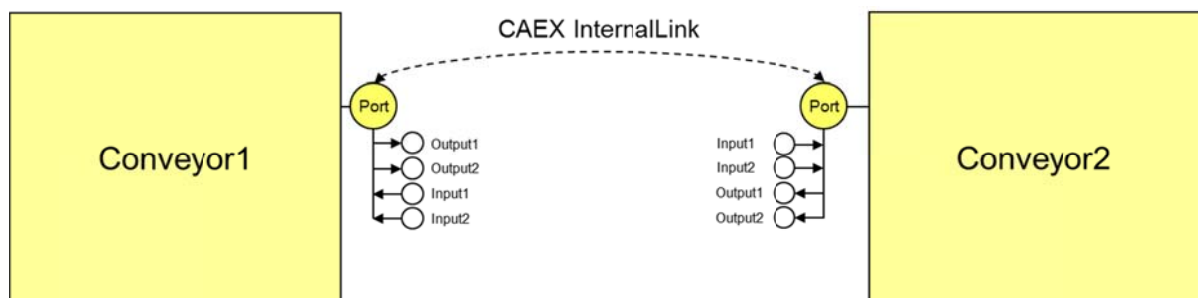


Figure A.2.1 – Port concept

#### A.2.2.2. Example

Figure A.2.2 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.

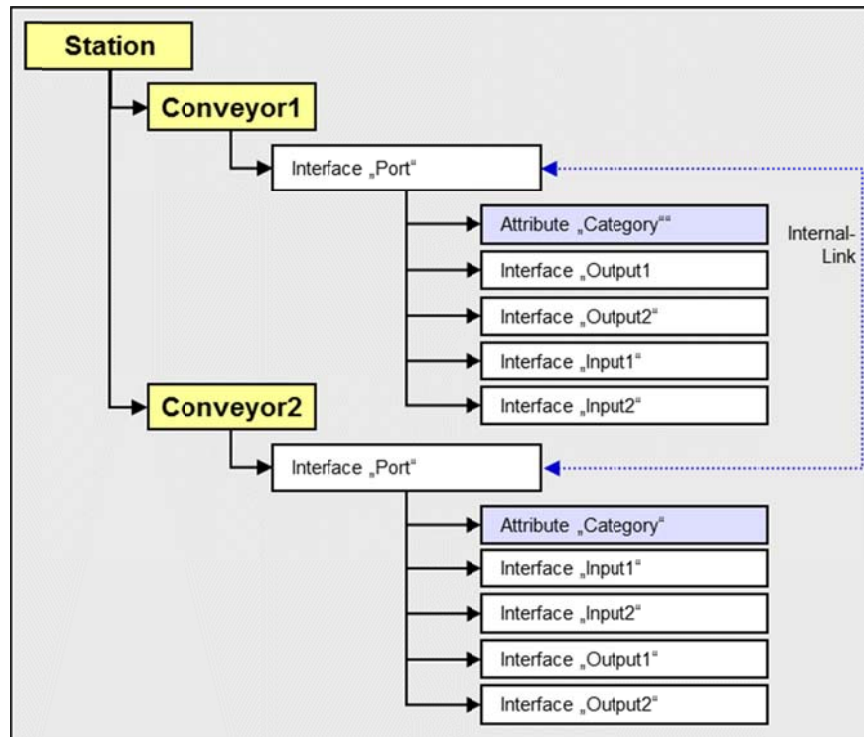


Figure A.2.2 – Example describing the AML Port concept

Figure A.2.3 and Figure A.2.4 describe the AML implementation of the example system described in Figure A.2.2.

InstanceHierarchy			
Name		PortExample AML 2.1	
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/SignaleInterface	GUID5
2	Output2	MyInterfaceClassLib/SignaleInterface	GUID6
3	Input1	MyInterfaceClassLib/SignaleInterface	GUID7
4	Input2	MyInterfaceClassLib/SignaleInterface	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/SignaleInterface	GUID12
2	Input2	MyInterfaceClassLib/SignaleInterface	GUID13
3	Output1	MyInterfaceClassLib/SignaleInterface	GUID14
4	Output2	MyInterfaceClassLib/SignaleInterface	GUID15
RoleRequirements			
RefBaseRoleClassPath		AutomationMLRoleClassLib/AutomationMLBaseRole/Resource	
InternalLink (1)			
	Name	RefPartnerSideA	RefPartnerSideB
1	L1	GUID3	GUID10

Figure A.2.3 – XML description of the AML Port concept

```

<InstanceHierarchy Name="PortExample AML 2.1">
  <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>

```

Figure A.2.4 – 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.2.5 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>

```

Figure A.2.5 – 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 whitepaper 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.2.6 explains the AML Facet concept by means of an example: the object “Conveyor1” comprises the attributes “A” and “B” as well as the interfaces “X” and “Y”. The assigned Facet object “PLCFacet” refers to the attribute “A” and the interface “X”, whereas the assigned Facet object “HMIFacet” refers to the attributes “A” and “B” as well as to the interface “Y”. Hence, both Facets provide a filtered view on certain engineering information which is relevant for different engineering tasks.

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

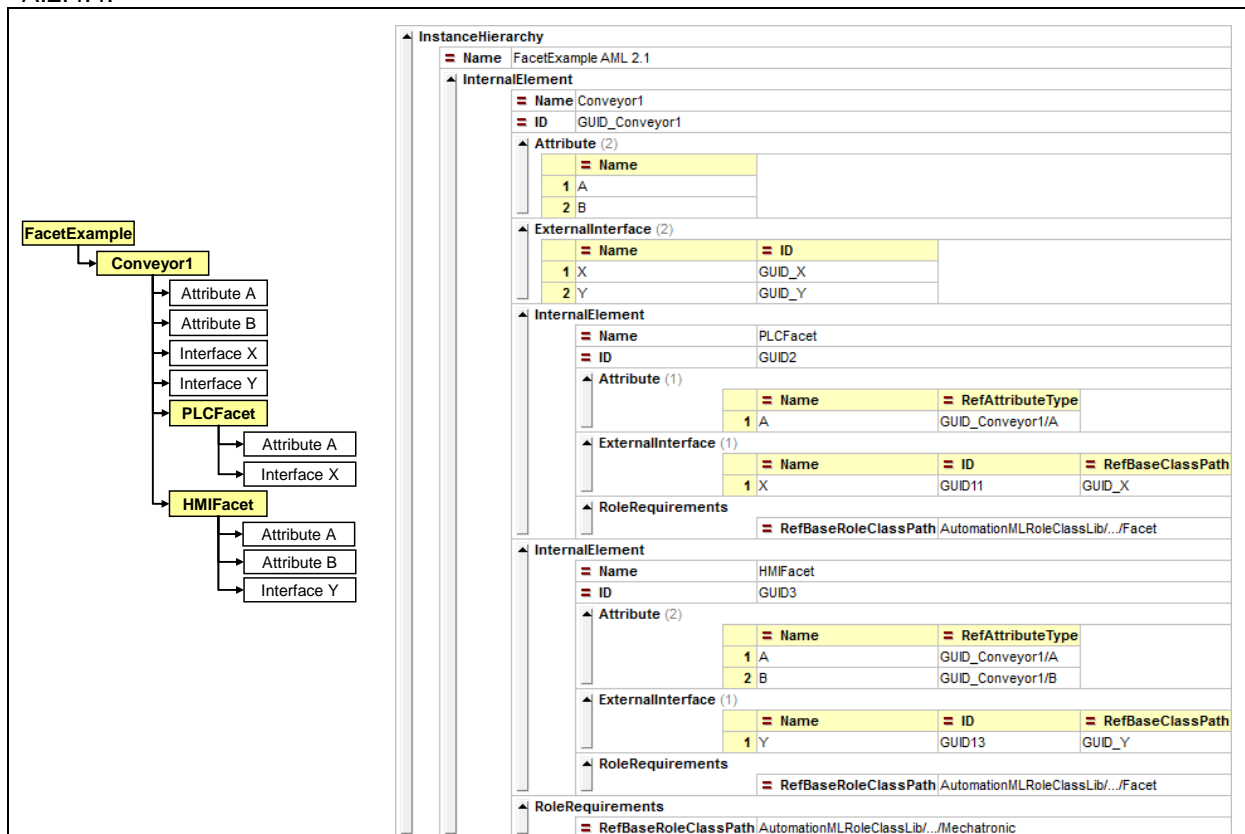


Figure A.2.6 – AML Facet example



```

<InstanceHierarchy Name="FacetExample AML 2.1">
  <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>

```

Figure A.2.7 – 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.2.8a) 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 62424:200862424:2015, A.2.14, CAEX provides the storage of such crossed structures. Figure A.2.8b) gives an AML implementation of this example and Figure A.2.9 provides the corresponding XML text. The combination between the Facet concept and the Port concept is described in A.2.4.3.

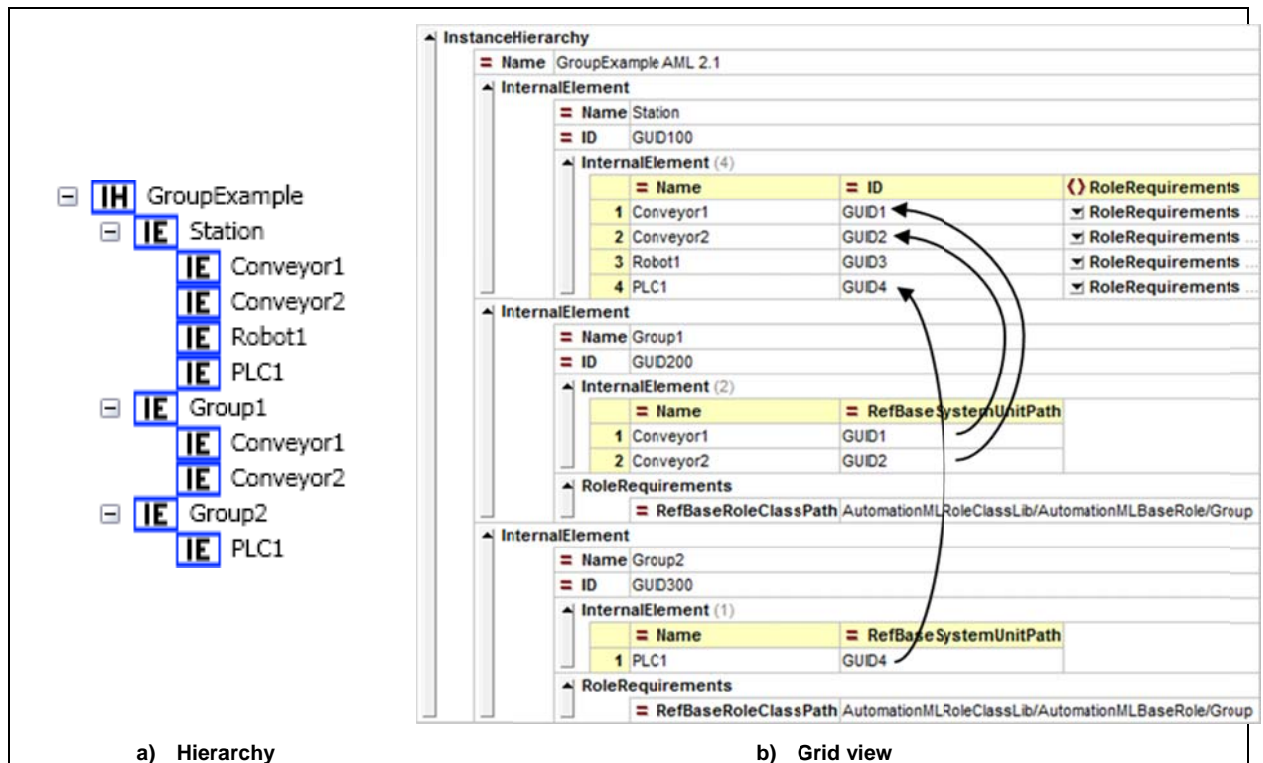


Figure A.2.8 – AML Group example

```
<InstanceHierarchy Name="GroupExample AML 2.1">
  <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>
```

Figure A.2.9 – XML text for the AML Group example

#### A.2.4.3. Combination of the Group and Facet concept

Figure A.2.10 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.

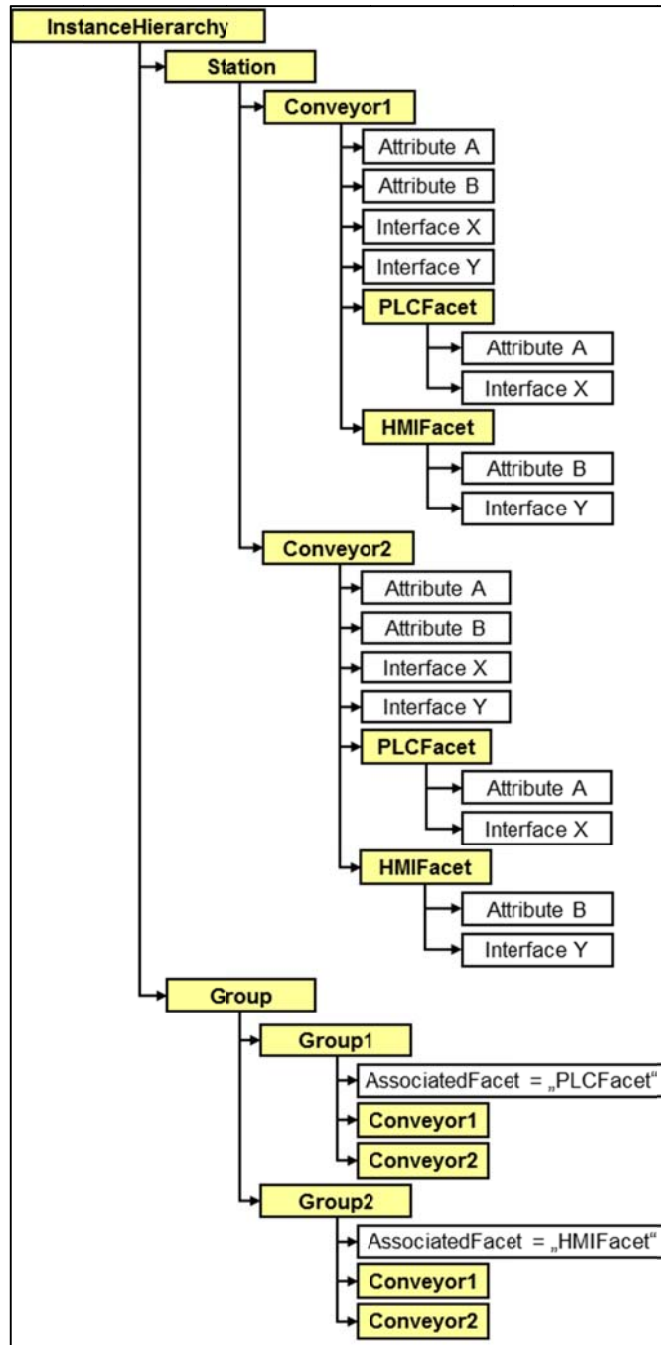


Figure A.2.10 – Combination of the Facet and Group concept

Figure A.2.11 shows the corresponding XML text related to the example shown in Figure A.2.10.



```

<InstanceHierarchy Name="FacetGroupCombination AML 2.1">
  <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>

```

Figure A.2.11 – 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.2.12 illustrates this generic HMI template of a conveyor.

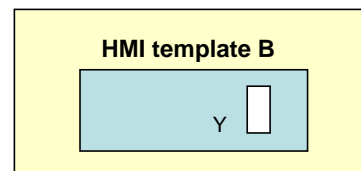


Figure A.2.12 – Generic HMI template "B" visualizing a process variable "Y" of a conveyor

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.

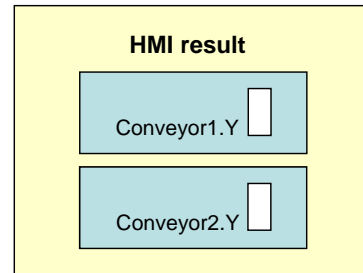


Figure A.2.13 represents the resulting HMI.

Figure A.2.13 – Generated HMI result “B” visualizing both conveyors with individual process variables

## A.2.5 Product-Process-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.2.14). 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.

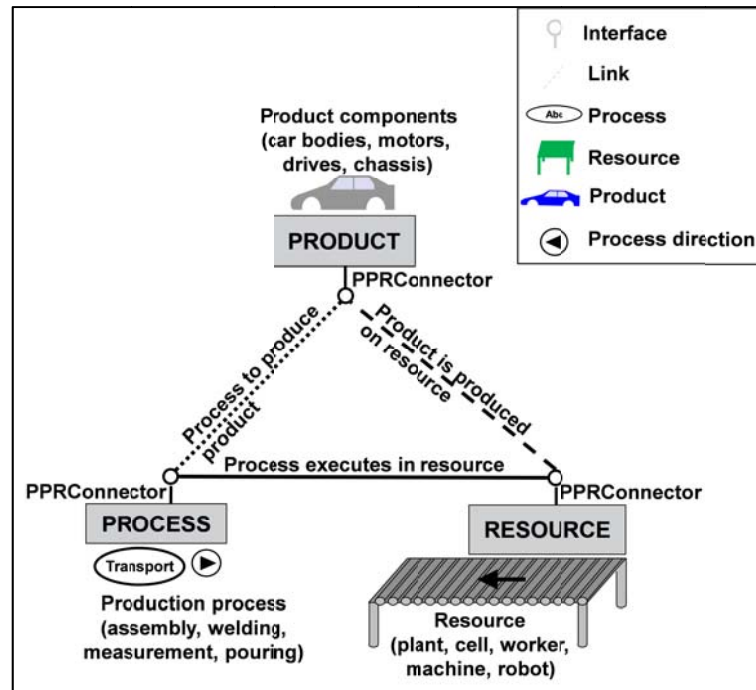


Figure A.2.14 – Base elements of the Product-Process-Resource concept

To create a link between these elements, they require an interface. For this, AML defines the standard interface class `PPRConnector` (see Figure A.2.15). 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.

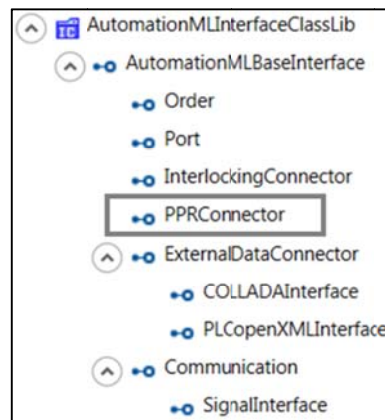


Figure A.2.15 – `PPRConnector` interface

#### A.2.5.2. Example

The following example (see Figure A.2.16) 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.

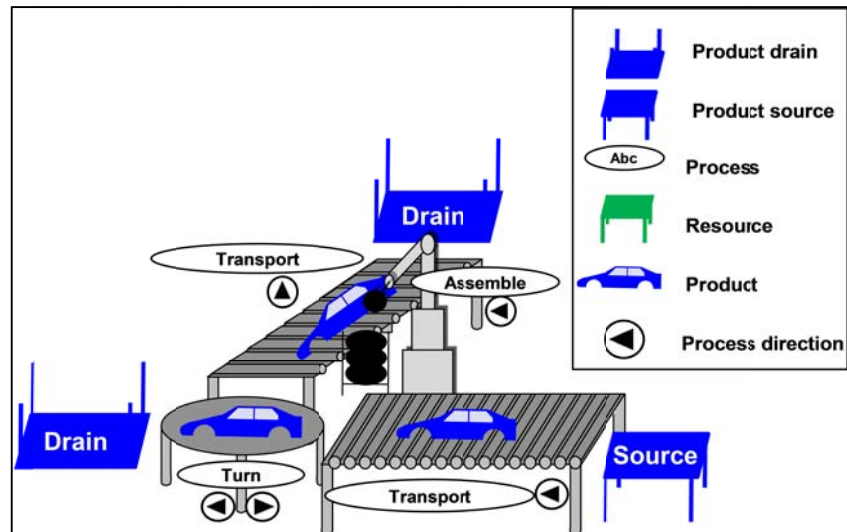


Figure A.2.16 – Example for the Product-Process-Resource concept

In AML, the Process-Product-Resource-concept is modelled by means of the CAEX role concept (see IEC 62424:2015, 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.2.17 and 6.4).

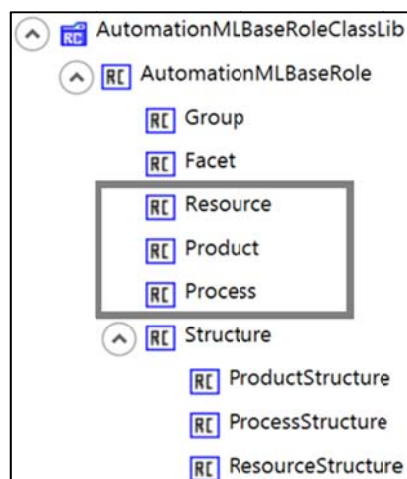


Figure A.2.17 – AML roles required for the Process-Product-Resource concept

In the example (see Figure A.2.16), 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.2.18. 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).

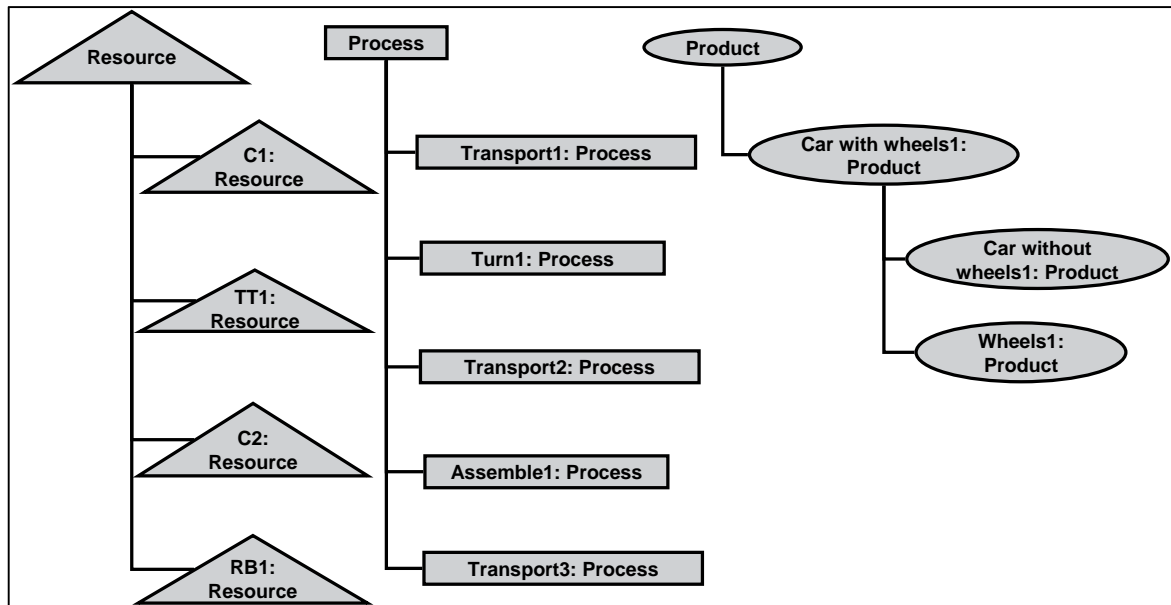


Figure A.2.18 – Elements of the example

Each element in the example has a PPRConnector interface. The complete links of the example are represented in Figure A.2.19. 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.

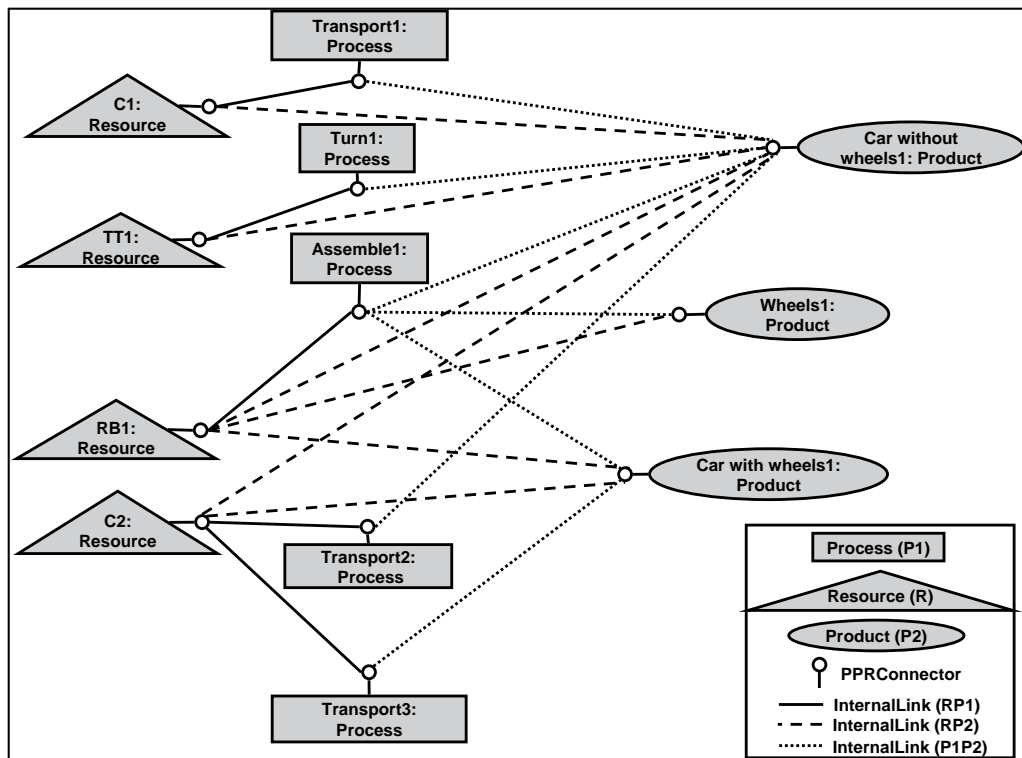


Figure A.2.19 – Links within the example

Figure A.2.20 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.2.19) can be derived from the existing links. The model can be arbitrarily rotated and arranged to centre the elements of type product or process.

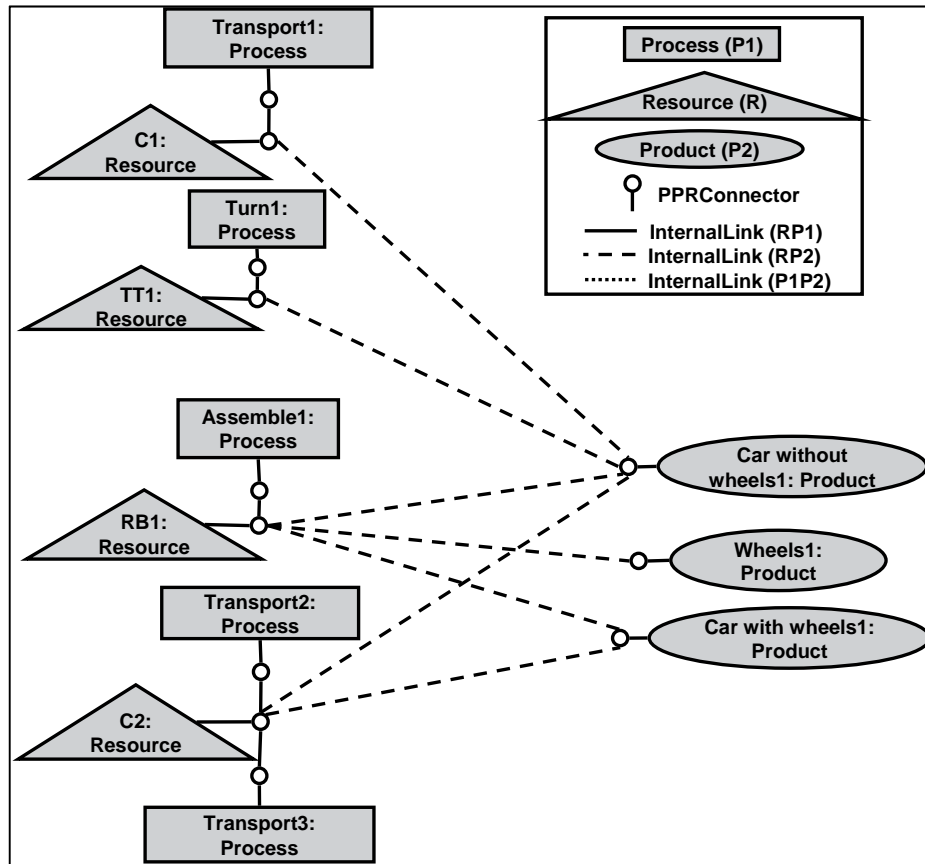


Figure A.2.20 – Links of the resource centric view on the example

Figure A.2.21 illustrates the AML object tree including a highlighted link between the conveyor “C1” and the process “Transport1”.

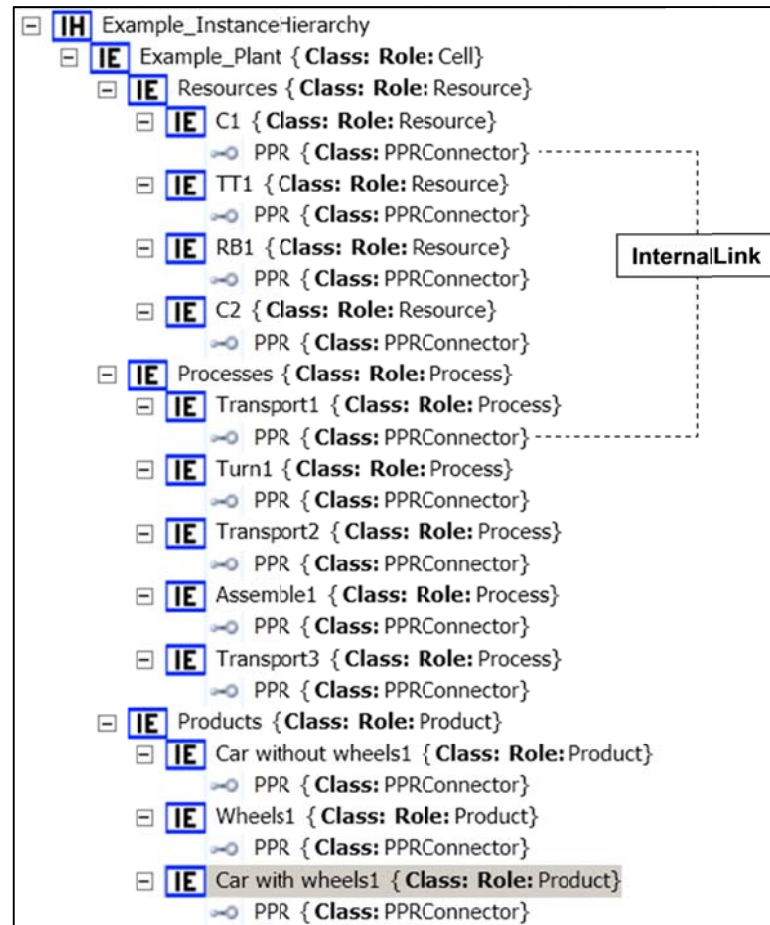


Figure A.2.21 – InstanceHierarchy of the example in AML

The corresponding XML model is depicted in Figure A.2.22. On the first level of the example, there are the three basic elements: “Resources”, “Processes” and “Products” modelled as CAEX InternalElement.



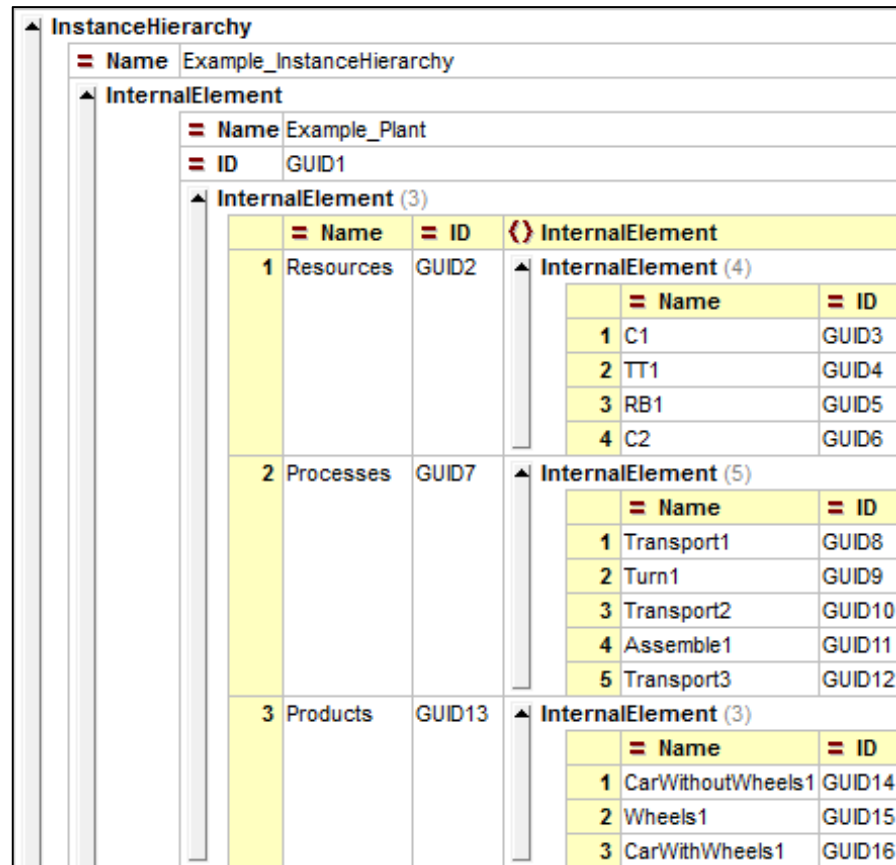


Figure A.2.22 – InternalElements of the example

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.2.23.

InternalLink (12)			
	= Name	= RefPartnerSideA	= RefPartnerSideB
1	C1_T1	GUID3_PPR	GUID8_PPR
2	TT1_Tu1	GUID4_PPR	GUID9_PPR
3	RB1_A1	GUID5_PPR	GUID11_PPR
4	C2_T2	GUID6_PPR	GUID10_PPR
5	C2_T3	GUID6_PPR	GUID12_PPR
6	C1_CwW1	GUID3_PPR	GUID14_PPR
7	TT1_CwW1	GUID4_PPR	GUID14_PPR
8	RB1_CwW1	GUID5_PPR	GUID14_PPR
9	RB1_W1	GUID5_PPR	GUID15_PPR
10	RB1_CW1	GUID5_PPR	GUID16_PPR
11	C2_CwW1	GUID6_PPR	GUID14_PPR
12	C2_CW1	GUID6_PPR	GUID16_PPR

Figure A.2.23 – InternalLinks of the example

A complete overview of the example can be seen in Figure A.2.24.



```

<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>

```

Figure A.2.24 – 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.2.25 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 RFC5654, their values contain the corresponding localized texts.

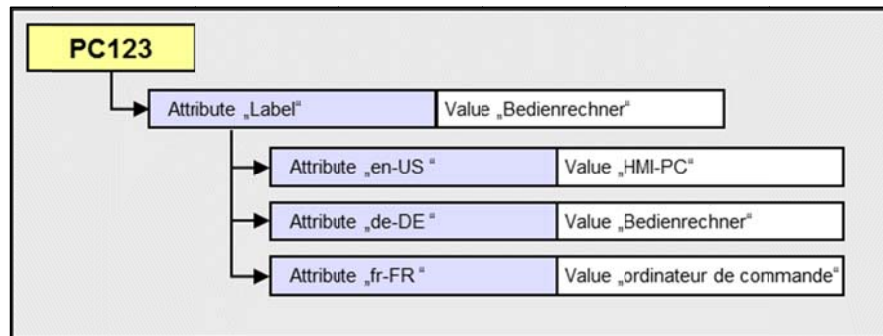


Figure A.2.25 – Example describing the AML multilingual expression concept

Figure A.2.26 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 RFC5654.	
() 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	

Figure A.2.26 – XML description of the AML multilingual expression concept

Figure A.2.27 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 RFC5654.</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>
  
```

Figure A.2.27 – 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. E.g., 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.2.28 illustrates this by the CAEX AttributeType *HMI-PC-LabelType*. It predefines all 3 localized language expressions.

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 RFC5654.	
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

Figure A.2.28 – AML model of a multilingual AttributeType

Figure A.2.29 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 RFC5654.</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>

```

Figure A.2.29 – 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. Example

The example shown in Figure A.2.30 exemplarily 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. Leading zeros are allowed and exemplarily added for better sortability.

InternalElement				
= Name		Radio		
= ID		GUID1		
Attribute				
= Name		SupportedFrequencies		
= RefAttributeType		AutomationMLBaseAttributeTypeLib/OrderedListType		
⌘ Description		a list of supported scan frequencies		
Attribute (4)				
	= Name	= Unit	= AttributeDataType	⌘ Value
1	0001	MHz	xs:float	90
2	0002	MHz	xs:float	95
3	0003	MHz	xs:float	100
4	0004	MHz	xs:float	105
RoleRequirements				
= RefBaseRoleClassPath		AutomationMLBaseRoleClassLib/.../Resource		

Figure A.2.30 – Attribute list “SupportedFrequencies”

Figure A.2.31 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>

```

Figure A.2.31 – XML code for the attribute list “SupportedFrequencies”

The example shown in Figure A.2.32 exemplarily 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)																																																																																							
		<table border="1"> <thead> <tr> <th></th> <th>= Name</th> <th>= RefAttributeType</th> <th>Attribute</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>EdgeNO</td> <td>AutomationMLBaseAttributeTypeLib/ListType</td> <td> <table border="1"> <thead> <tr> <th></th> <th>= Name</th> <th>= AttributeDataType</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>x</td> <td>xs:float</td> <td>10</td> </tr> <tr> <td>2</td> <td>y</td> <td>xs:float</td> <td>10</td> </tr> <tr> <td>3</td> <td>z</td> <td>xs:float</td> <td>10</td> </tr> </tbody> </table> </td> </tr> <tr> <td>2</td> <td>EdgeSO</td> <td>AutomationMLBaseAttributeTypeLib/ListType</td> <td> <table border="1"> <thead> <tr> <th></th> <th>= Name</th> <th>= AttributeDataType</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>x</td> <td>xs:float</td> <td>10</td> </tr> <tr> <td>2</td> <td>y</td> <td>xs:float</td> <td>0</td> </tr> <tr> <td>3</td> <td>z</td> <td>xs:float</td> <td>10</td> </tr> </tbody> </table> </td> </tr> <tr> <td>3</td> <td>EdgeSW</td> <td>AutomationMLBaseAttributeTypeLib/ListType</td> <td> <table border="1"> <thead> <tr> <th></th> <th>= Name</th> <th>= AttributeDataType</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>x</td> <td>xs:float</td> <td>0</td> </tr> <tr> <td>2</td> <td>y</td> <td>xs:float</td> <td>0</td> </tr> <tr> <td>3</td> <td>z</td> <td>xs:float</td> <td>10</td> </tr> </tbody> </table> </td> </tr> <tr> <td>4</td> <td>EdgeNW</td> <td>AutomationMLBaseAttributeTypeLib/ListType</td> <td> <table border="1"> <thead> <tr> <th></th> <th>= Name</th> <th>= AttributeDataType</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>x</td> <td>xs:float</td> <td>0</td> </tr> <tr> <td>2</td> <td>y</td> <td>xs:float</td> <td>10</td> </tr> <tr> <td>3</td> <td>z</td> <td>xs:float</td> <td>10</td> </tr> </tbody> </table> </td> </tr> </tbody> </table>		= Name	= RefAttributeType	Attribute	1	EdgeNO	AutomationMLBaseAttributeTypeLib/ListType	<table border="1"> <thead> <tr> <th></th> <th>= Name</th> <th>= AttributeDataType</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>x</td> <td>xs:float</td> <td>10</td> </tr> <tr> <td>2</td> <td>y</td> <td>xs:float</td> <td>10</td> </tr> <tr> <td>3</td> <td>z</td> <td>xs:float</td> <td>10</td> </tr> </tbody> </table>		= Name	= AttributeDataType	Value	1	x	xs:float	10	2	y	xs:float	10	3	z	xs:float	10	2	EdgeSO	AutomationMLBaseAttributeTypeLib/ListType	<table border="1"> <thead> <tr> <th></th> <th>= Name</th> <th>= AttributeDataType</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>x</td> <td>xs:float</td> <td>10</td> </tr> <tr> <td>2</td> <td>y</td> <td>xs:float</td> <td>0</td> </tr> <tr> <td>3</td> <td>z</td> <td>xs:float</td> <td>10</td> </tr> </tbody> </table>		= Name	= AttributeDataType	Value	1	x	xs:float	10	2	y	xs:float	0	3	z	xs:float	10	3	EdgeSW	AutomationMLBaseAttributeTypeLib/ListType	<table border="1"> <thead> <tr> <th></th> <th>= Name</th> <th>= AttributeDataType</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>x</td> <td>xs:float</td> <td>0</td> </tr> <tr> <td>2</td> <td>y</td> <td>xs:float</td> <td>0</td> </tr> <tr> <td>3</td> <td>z</td> <td>xs:float</td> <td>10</td> </tr> </tbody> </table>		= Name	= AttributeDataType	Value	1	x	xs:float	0	2	y	xs:float	0	3	z	xs:float	10	4	EdgeNW	AutomationMLBaseAttributeTypeLib/ListType	<table border="1"> <thead> <tr> <th></th> <th>= Name</th> <th>= AttributeDataType</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>x</td> <td>xs:float</td> <td>0</td> </tr> <tr> <td>2</td> <td>y</td> <td>xs:float</td> <td>10</td> </tr> <tr> <td>3</td> <td>z</td> <td>xs:float</td> <td>10</td> </tr> </tbody> </table>		= Name	= AttributeDataType	Value	1	x	xs:float	0	2	y	xs:float	10	3	z	xs:float	10	
	= Name	= RefAttributeType	Attribute																																																																																				
1	EdgeNO	AutomationMLBaseAttributeTypeLib/ListType	<table border="1"> <thead> <tr> <th></th> <th>= Name</th> <th>= AttributeDataType</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>x</td> <td>xs:float</td> <td>10</td> </tr> <tr> <td>2</td> <td>y</td> <td>xs:float</td> <td>10</td> </tr> <tr> <td>3</td> <td>z</td> <td>xs:float</td> <td>10</td> </tr> </tbody> </table>		= Name	= AttributeDataType	Value	1	x	xs:float	10	2	y	xs:float	10	3	z	xs:float	10																																																																				
	= Name	= AttributeDataType	Value																																																																																				
1	x	xs:float	10																																																																																				
2	y	xs:float	10																																																																																				
3	z	xs:float	10																																																																																				
2	EdgeSO	AutomationMLBaseAttributeTypeLib/ListType	<table border="1"> <thead> <tr> <th></th> <th>= Name</th> <th>= AttributeDataType</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>x</td> <td>xs:float</td> <td>10</td> </tr> <tr> <td>2</td> <td>y</td> <td>xs:float</td> <td>0</td> </tr> <tr> <td>3</td> <td>z</td> <td>xs:float</td> <td>10</td> </tr> </tbody> </table>		= Name	= AttributeDataType	Value	1	x	xs:float	10	2	y	xs:float	0	3	z	xs:float	10																																																																				
	= Name	= AttributeDataType	Value																																																																																				
1	x	xs:float	10																																																																																				
2	y	xs:float	0																																																																																				
3	z	xs:float	10																																																																																				
3	EdgeSW	AutomationMLBaseAttributeTypeLib/ListType	<table border="1"> <thead> <tr> <th></th> <th>= Name</th> <th>= AttributeDataType</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>x</td> <td>xs:float</td> <td>0</td> </tr> <tr> <td>2</td> <td>y</td> <td>xs:float</td> <td>0</td> </tr> <tr> <td>3</td> <td>z</td> <td>xs:float</td> <td>10</td> </tr> </tbody> </table>		= Name	= AttributeDataType	Value	1	x	xs:float	0	2	y	xs:float	0	3	z	xs:float	10																																																																				
	= Name	= AttributeDataType	Value																																																																																				
1	x	xs:float	0																																																																																				
2	y	xs:float	0																																																																																				
3	z	xs:float	10																																																																																				
4	EdgeNW	AutomationMLBaseAttributeTypeLib/ListType	<table border="1"> <thead> <tr> <th></th> <th>= Name</th> <th>= AttributeDataType</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>x</td> <td>xs:float</td> <td>0</td> </tr> <tr> <td>2</td> <td>y</td> <td>xs:float</td> <td>10</td> </tr> <tr> <td>3</td> <td>z</td> <td>xs:float</td> <td>10</td> </tr> </tbody> </table>		= Name	= AttributeDataType	Value	1	x	xs:float	0	2	y	xs:float	10	3	z	xs:float	10																																																																				
	= Name	= AttributeDataType	Value																																																																																				
1	x	xs:float	0																																																																																				
2	y	xs:float	10																																																																																				
3	z	xs:float	10																																																																																				
RoleRequirements																																																																																							
= RefBaseRoleClassPath		AutomationMLBaseRoleClassLib/.../Resource																																																																																					

Figure A.2.32 – Example CAEX model of the array “Edges”

Figure A.2.33 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>

```

Figure A.2.33 - XML code for the attribute array "Edges"



## B. XML Representation of standard AML base libraries

```

<CAEXFile xmlns="http://www.dke.de/CAEX" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" SchemaVersion="3.0" FileName="
AutomationML2.1BaseLibraries" xsi:schemaLocation="http://www.dke.de/CAEX CAEX_ClassModel_V.3.0.xsd">
  <SuperiorStandardVersion>AutomationML 2.1</SuperiorStandardVersion>
  <SourceDocumentInformation OriginID="IEC SC65E WG 9" OriginName="IEC SC65E WG 9" OriginVersion="2.1.0" LastWritingDateTime="
2015-10-06T14:43:00.0Z" OriginProjectID="Automation Markup Language Standard Library" OriginRelease="2.1.0" 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>2.2.5</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>
      <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>2.2.5</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>1.0</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="AssociatedValue"/>
    <AttributeType Name="MIMEType" AttributeDataType="xs:string"/>
    <AttributeType Name="DocLang" AttributeDataType="xs:string"/>
  </AttributeTypeLib>
</CAEXFile>

```

Figure B.1 - InstanceHierarchy of the example in XML



**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*

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