BS EN 62714-1:2014

BSI Standards Publication

Engineering data exchange format for use in industrial automation systems engineering

Part 1: Architecture and General Requirements

bsi.

...making excellence a habit.™

National foreword

This British Standard is the UK implementation of EN 62714-1:2014. It is identical to IEC 62714-1:2014.

The UK participation in its preparation was entrusted to Technical Committee AMT/7, Industrial communications: process measurement and control, including fieldbus.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

Published by BSI Standards Limited 2014

ISBN 978 0 580 73983 5

ICS 25.040.40; 35.060; 35.240.50

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 October 2014.

Amendments/corrigenda issued since publication

Date	Text affected

EUROPEAN STANDARD

NORME EUROPÉENNE

EUROPÄISCHE NORM

EN 62714-1

October 2014

ICS 25.040.40; 35.060; 35.240.50

English Version

Engineering data exchange format for use in industrial automation systems engineering - Part 1: Architecture and General Requirements (IEC 62714-1:2014)

Format d'échange de données techniques pour une utilisation dans l'ingénierie des systèmes d'automatisation industrielle - AutomationML - Partie 1: Architecture et exigences générales
(CEI 62714-1:2014)

Datenaustauschformat für Planungsdaten industrieller Automatisierungssysteme (AutomationML) - Teil 1: Architektur und allgemeine Festlegungen
(IEC 62714-1:2014)

This European Standard was approved by CENELEC on 2014-07-31. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the CEN-CENELEC Management Centre or to any CENELEC member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CENELEC member into its own language and notified to the CEN-CENELEC Management Centre has the same status as the official versions.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Bulgaria, Croatia, Cyprus, the Czech Republic, Denmark, Estonia, Finland, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and the United Kingdom.

CENELEC

European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

CEN-CENELEC Management Centre: Avenue Marnix 17, B-1000 Brussels

Ref. No. EN 62714-1:2014 E

Foreword

The text of document 65E/385/FDIS, future edition 1 of IEC 62714-1, prepared by SC 65E "Devices and integration in enterprise systems" of IEC/TC 65 "Industrial-process measurement, control and automation" was submitted to the IEC-CENELEC parallel vote and approved by CENELEC as EN 62714-1:2014.

The following dates are fixed:

• latest date by which the document has to be implemented at national level by publication of an identical national standard or by endorsement	(dop)	2015-05-01
• latest date by which the national standards conflicting with the document have to be withdrawn	(dow)	2017-07-31

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CENELEC [and/or CEN] shall not be held responsible for identifying any or all such patent rights.

This document has been prepared under a mandate given to CENELEC by the European Commission and the European Free Trade Association.

Endorsement notice

The text of the International Standard IEC 62714-1:2014 was approved by CENELEC as a European Standard without any modification.

In the official version, for Bibliography, the following notes have to be added for the standards indicated:

IEC 60027 (Series)	NOTE	Harmonized as EN 60027 (Series).
IEC 62264-1	NOTE	Harmonized as EN 62264-1.
IEC 62714-2	NOTE	Harmonized as EN 62714-2
ISO 80000-1	NOTE	Harmonized as EN ISO 80000-1.

Annex ZA
(normative)

Normative references to international publications with their corresponding European publications

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

NOTE 1 When an International Publication has been modified by common modifications, indicated by (mod), the relevant EN/HD applies.

NOTE 2 Up-to-date information on the latest versions of the European Standards listed in this annex is available here: www.cenelec.eu.

Publication	Year	Title	EN/HD	Year
IEC 62424	2008	Representation of process control engineering - Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools	EN 62424	2009
IEC 62714	series	Engineering data exchange format for use in industrial automation systems engineering	EN 62714	series
ISO/IEC 9834-8	-	Information technology - Procedures for the operation of object identifier registration authorities: General procedures and top arcs of the international object identifier tree	-	-
ISO/PAS 17506	-	Industrial automation systems and integration - COLLADA digital asset schema specification for 3D visualization of industrial data	-	-

CONTENTS

INTRODUCTION ... 9
1 Scope ... 11
2 Normative references ... 11
3 Terms, definitions and abbreviations ... 11
3.1 Terms and definitions ... 11
3.2 Abbreviations ... 14
4 Conformity ... 14
5 AML architecture specification ... 15
5.1 General ... 15
5.2 General AML architecture ... 15
5.3 AML document versions ... 16
5.4 Meta information about the AML source tool ... 17
5.5 Object identification ... 18
5.6 AML relations specification ... 19
5.6.1 General ... 19
5.6.2 Parent-child-relations between AML objects ... 19
5.6.3 Parent-child-relations between AML classes ... 20
5.6.4 Inheritance relations ... 21
5.6.5 Class-instance-relations ... 21
5.6.6 Instance-instance-relations ... 23
5.7 AML document reference specification ... 25
5.7.1 General ... 25
5.7.2 Referencing COLLADA documents ... 25
5.7.3 Referencing PLCopen XML documents ... 25
5.7.4 Referencing additional documents ... 25
6 AML base libraries ... 25
6.1 General ... 25
6.2 General provisions ... 25
6.3 AML interface class library – AutomationMLInterfaceClassLib ... 26
6.3.1 General ... 26
6.3.2 InterfaceClass AutomationMLBaseInterface ... 28
6.3.3 InterfaceClass Order ... 28
6.3.4 InterfaceClass PortConnector ... 29
6.3.5 InterfaceClass PPRConnector ... 29
6.3.6 InterfaceClass ExternalDataConnector ... 29
6.3.7 InterfaceClass COLLADAInterface ... 30
6.3.8 InterfaceClass PLCopenXMLInterface ... 30
6.3.9 InterfaceClass Communication ... 30
6.3.10 InterfaceClass SignalInterface ... 31
6.4 AML basic role class library – AutomationMLBaseRoleClassLib ... 31
6.4.1 General ... 31
6.4.2 RoleClass AutomationMLBaseRole ... 33
6.4.3 RoleClass Group ... 33
6.4.4 RoleClass Facet ... 34

6.4.5 RoleClass Port 34
6.4.6 RoleClass Resource 36
6.4.7 RoleClass Product 36
6.4.8 RoleClass Process 37
6.4.9 RoleClass Structure 37
6.4.10 RoleClass ProductStructure 37
6.4.11 RoleClass ProcessStructure 38
6.4.12 RoleClass ResourceStructure 38
6.4.13 RoleClass PropertySet 38
7 Modelling of user-defined data 39
7.1 General 39
7.2 User-defined attributes 39
7.3 User-defined InterfaceClasses 39
7.4 User-defined RoleClasses 40
7.5 User-defined SystemUnitClasses 41
7.6 User-defined InstanceHierarchies 41
8 Extended AML concepts 42
8.1 General overview 42
8.2 AML Port object 42
8.3 AML Facet object 43
8.4 AML Group object 43
8.5 AML PropertySet 44
8.6 Support of multiple roles 46
8.7 Splitting of AML top-level data into different documents 47
8.8 Internationalization 47
8.9 Version information of AML objects 47
Annex A (informative) 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 50
A.1.3 Referencing geometry and kinematics information 51
A.1.4 Referencing logic information 51
A.1.5 Modelling of relations 52
A.2 Extended AML concepts and examples 55
A.2.1 General overview 55
A.2.2 AML Port concept 55
A.2.3 AML Facet concept 59
A.2.4 AML Group concept 61
A.2.5 PropertySet concept 65
A.2.6 Process-Product-Resource concept 68
A.2.7 Support of multiple roles 76
Annex B (informative) XML Representation of AML Libraries 80
B.1 AutomationMLBaseRoleClassLib 80
B.2 AutomationMLInterfaceClassLib 81
Bibliography 82

Figure 1 – Overview of the engineering data exchange format AML 9
Figure 2 – AML document version information 16

Figure 3 – XML text of the AML source tool information 18
Figure 4 – Object identification example of an AML class 19
Figure 5 – Object identification example of an AML object instance 19
Figure 6 – Example of a parent-child-relation between AML objects 20
Figure 7 – Example of a parent-child-relation between classes 20
Figure 8 – Example of an inheritance relation between two classes 21
Figure 9 – Example of a class-instance-relation 22
Figure 10 – Example of a relation as block diagram and as object tree 23
Figure 11 – Example relation between the objects “PLC1” and “Rob1” 24
Figure 12 – AML basic interface class library 27
Figure 13 – XML description of the AML basic interface class library 28
Figure 14 – AML basic role class library 32
Figure 15 – AutomationMLBaseRoleClassLib 32
Figure 16 – XML text of the AutomationMLBaseRoleClassLib 33
Figure 17 – Example of a user-defined attribute 39
Figure 18 – Example of a user-defined InterfaceClass in a user-defined InterfaceClassLib 40
Figure 19 – Example of a user-defined RoleClass in a user-defined RoleClassLib 41
Figure 20 – Examples for different user-defined SystemUnitClasses 41
Figure 21 – Example of a user-defined InstanceHierarchy 42
Figure 22 – AML representation of a user-defined InstanceHierarchy 42
Figure 23 – Example illustrating the PropertySet concept 45
Figure 24 – XML text of the PropertySet example 46
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 52
Figure A.5 – Relations in AML 53
Figure A.6 – XML description of the relations example 54
Figure A.7 – XML text of the SystemUnitClassLib of the relations example 54
Figure A.8 – XML text of the InstanceHierarchy of the relations example 54
Figure A.9 – Port concept 55
Figure A.10 – Example describing the AML Port concept 56
Figure A.11 – XML description of the AML Port concept 57
Figure A.12 – XML text describing the AML Port concept 58
Figure A.13 – Definition of a user-defined AML Port class “myPortClass” 58
Figure A.14 – AML Facet example 60
Figure A.15 – XML text of the AML Facet example 60
Figure A.16 – AML Group example 61
Figure A.17 – XML text for the AML Group example 62
Figure A.18 – Combination of the Facet and Group concept 63
Figure A.19 – XML text view for the combined Facet-Group example 64
Figure A.20 – Generic HMI template “B” visualizing a process variable “Y” of a conveyor 65

Figure A.21 – Generated HMI result “B” visualizing both conveyors with individual process variables 65
Figure A.22 – PropertySet example 66
Figure A.23 – PropertySet example 66
Figure A.24 – XML text for the instance hierarchy 67
Figure A.25 – PropertySet example AML library as XML code 68
Figure A.26 – Base elements of the Product-Process-Resource concept 69
Figure A.27 – PPRConnector interface 70
Figure A.28 – Example for the Product-Process-Resource concept 70
Figure A.29 – AML roles required for the Process-Product-Resource concept 71
Figure A.30 – Elements of the example 71
Figure A.31 – Links within the example 72
Figure A.32 – Links of the resource centric view on the example 73
Figure A.33 – InstanceHierarchy of the example in AML 74
Figure A.34 – InternalElements of the example 75
Figure A.35 – InternalLinks of the example 75
Figure A.36 – InstanceHierarchy of the example in XML 76
Figure A.37 – Example of a user-defined instance supporting multiple roles 77
Figure A.38 – XML text of the AML representation of multiple role support 78
Figure A.39 – AML Role class library corresponding to the multiple role definition example 78
Figure A.40 – XML text of the AML role class library 79

Table 1 – Abbreviations 14
Table 2 – Meta information about the AML source tool 17
Table 3 – Interface classes of the AutomationMLInterfaceClassLib 26
Table 4 – InterfaceClass AutomationMLBaseInterface 28
Table 5 – InterfaceClass Order 28
Table 6 – InterfaceClass PortConnector 29
Table 7 – InterfaceClass PPRConnector 29
Table 8 – InterfaceClass ExternalDataConnector 29
Table 9 – InterfaceClass COLLADAInterface 30
Table 10 – InterfaceClass PLCopenXMLInterface 30
Table 11 – InterfaceClass Communication 31
Table 12 – InterfaceClass SignalInterface 31
Table 13 – RoleClass AutomationMLBaseRole 33
Table 14 – RoleClass Group 34
Table 15 – RoleClass Facet 34
Table 16 – Optional attributes for AML Port objects 35
Table 17 – Sub-attributes of the attribute “Cardinality” 35
Table 18 – Interface of the AML Port class 36
Table 19 – RoleClass Resource 36
Table 20 – RoleClass Product 36
Table 21 – RoleClass Process 37

Table 22 – RoleClass Structure 37
Table 23 – RoleClass ProductStructure 37
Table 24 – RoleClass ProcessStructure 38
Table 25 – RoleClass ResourceStructure 38
Table 26 – RoleClass PropertySet 38
Table A.1 – Overview of major extended AML concepts 55

INTRODUCTION

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

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

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

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

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

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

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

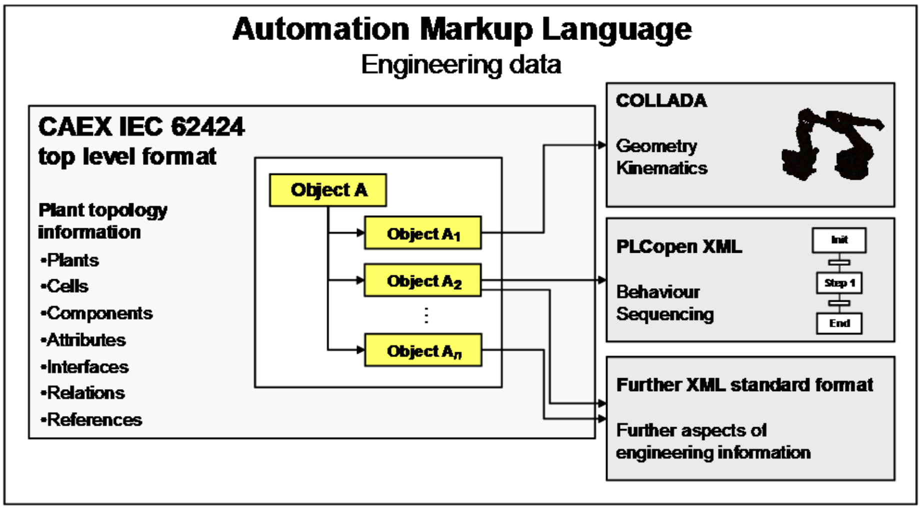


Figure 1 – Overview of the engineering data exchange format AML

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

- IEC 62714-1: Architecture and general requirements

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

- IEC 62714-2: Role class libraries

  This part is intended to specify additional AML libraries.
- IEC 62714-3: Geometry and kinematics

  This part is intended to specify the modelling of geometry and kinematics information.
- IEC 62714-4: Logic

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

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

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

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

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

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

Part 1: Architecture and general requirements

1 Scope

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

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

2 Normative references

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

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

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

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

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

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

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

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

3 Terms, definitions and abbreviations

3.1 Terms and definitions

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

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

3.1.2
automation object
physical or logical entity in the automated system

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

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

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

3.1.4
AML class
predefined AML object type

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

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

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

3.1.5
AML attribute
property which belongs to an AML object

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

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

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

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

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

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

3.1.9
AML library
library containing AML classes

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

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

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

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

3.1.13
CAEX
neutral XML based data format

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

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

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

3.1.15
universal unique identifier
UUID
unique identifier for AML objects

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

3.1.16
global unique identifier
GUID
implementation of a UUID

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

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

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

3.1.17
inheritance relation
relation between two AML classes

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

3.1.18
instance
data representation of an individual physical or logical item

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

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

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

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

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

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

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

3.1.23
relation
association between CAEX objects

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

3.1.24
link
connection between objects of type CAEX ExternalInterface

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

3.1.25
reference
association between a CAEX InternalElement and externally stored information

3.2 Abbreviations

Table 1 – Abbreviations

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

4 Conformity

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

5 AML architecture specification

5.1 General

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

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

5.2 General AML architecture

Regarding the general AML architecture, the following provisions apply:

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

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

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

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

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

NOTE 4 Relations depict associations between CAEX objects.

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

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

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

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

---

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

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

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

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

5.3 AML document versions

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

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

Hence, the following provisions apply:

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

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

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

Figure 2 – AML document version information

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

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

IEC 62714 is based on the following document formats:

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

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

5.4 Meta information about the AML source tool

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

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

Table 2 – Meta information about the AML source tool

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

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

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

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

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

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

Figure 3 – XML text of the AML source tool information

5.5 Object identification

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

Regarding the object identification, the following provisions apply:

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

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

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

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

  NOTE 4 The CAEX tag “Name” is a display name; it has informative character only and can change over the time or tool.

- Once created, this UUID shall never change over the life time of the corresponding object within all participating tools.
- Referencing instances shall use the “ID” value.
- Referencing CAEX interfaces shall be done using the corresponding UUID of the interface’s parent object followed by the separator string “:”and the name of the interface instance.

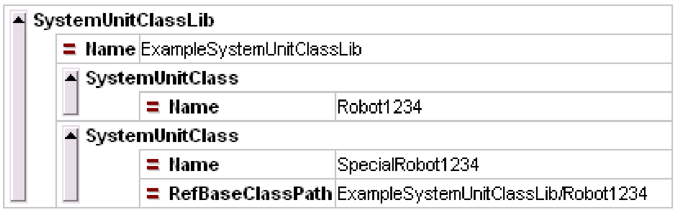
EXAMPLE 1 “GUID:out”.

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

EXAMPLE 2 “GUID.Colour”.

EXAMPLE 3 “GUID.Colour.red”.

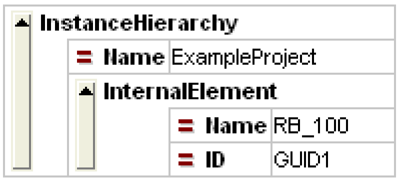
Figure 4 gives an example of a SystemUnitClassLib with the SystemUnitClass “Robot1234” and another SystemUnitClass “SpecialRobot1234” derived from “Robot1234”.




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

Figure 4 – Object identification example of an AML class

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




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

Figure 5 – Object identification example of an AML object instance

5.6 AML relations specification

5.6.1 General

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

5.6.2 Parent-child-relations between AML objects

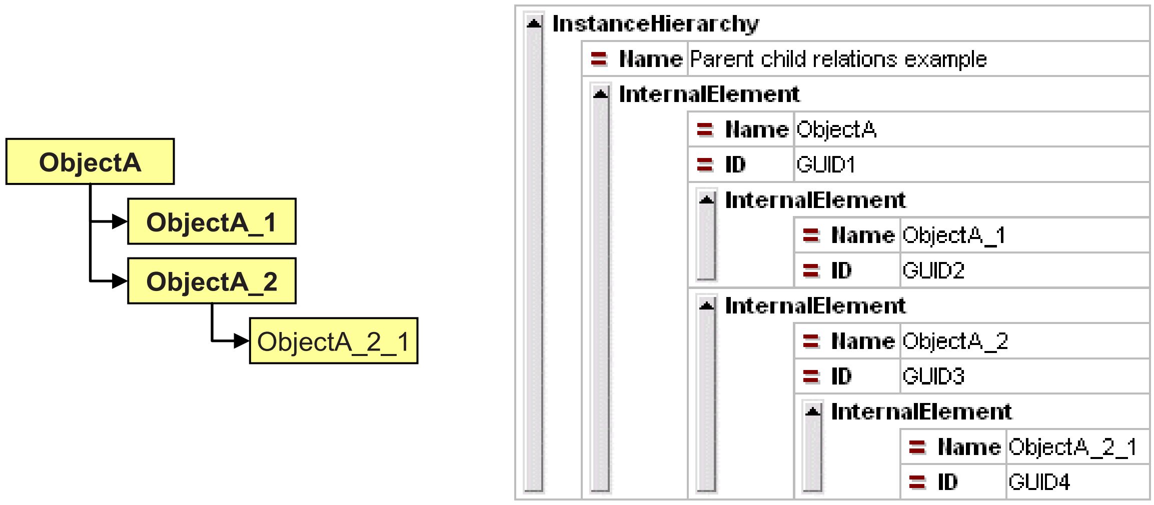
Parent-child-relations between object instances are used to represent hierarchical object structures and describe a “consist-of-relation”.

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

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

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

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




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

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

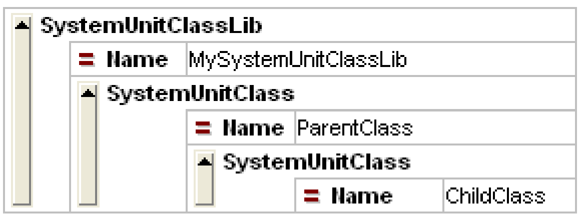
5.6.3 Parent-child-relations between AML classes

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

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

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

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




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

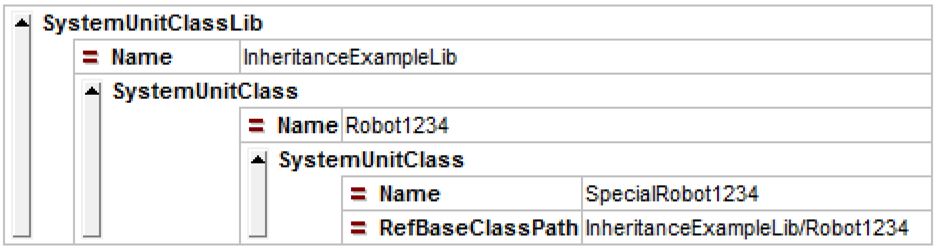
Figure 7 – Example of a parent-child-relation between classes

5.6.4 Inheritance relations

Regarding inheritance relations, the following provisions apply:

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

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




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

Figure 8 – Example of an inheritance relation between two classes

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

5.6.5 Class-instance-relations

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

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

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

  NOTE 2 Instances without a relation to the AutomationMLBaseRole are possible but are user defined objects. They are not AML objects.
- If an AML object has a class-instance-relation to a SystemUnitClass, it shall be created as a copy of this SystemUnitClass including the internal architecture of the class and all inherited information.

  NOTE 3 A SystemUnitClass serves as a template in this way. Changes in the SystemUnitClass are not automatically reflected in the corresponding Automation objects. Furthermore, the Automation object can be transported without the class information; it contains within itself the whole belonging information.
- The extension or reduction of instance data compared to the source class is allowed.

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

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

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

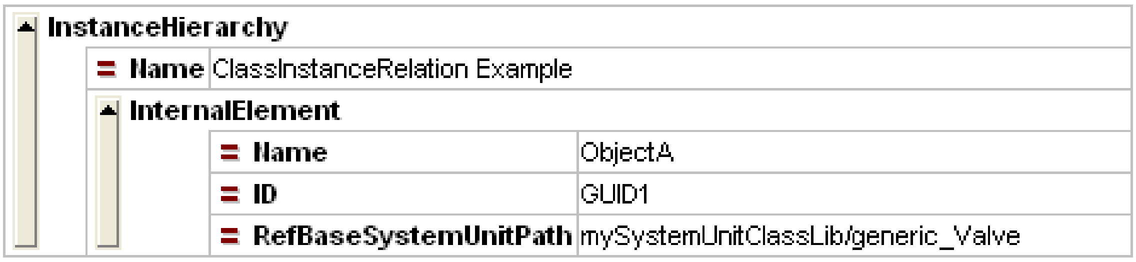
  NOTE 6 This provision supports tracking of changes across different versions of a class.

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

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

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

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




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

Figure 9 – Example of a class-instance-relation

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

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

5.6.6 Instance-instance-relations

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

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

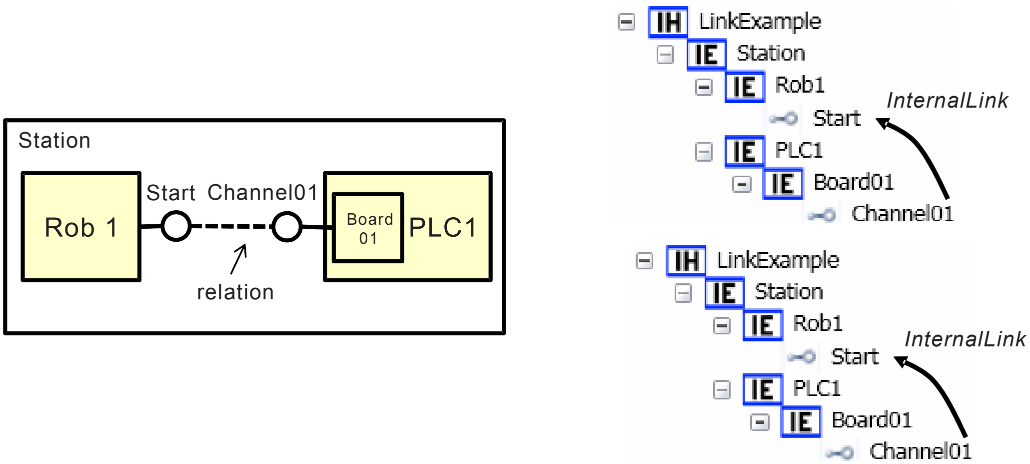
- Instance-instance-relations shall be stored according to IEC 62424:2008, A.2.5.3 and A.2.14, by means of the CAEX InternalLink functionality.
- CAEX InternalLinks should be stored at the CAEX InternalElement which is the lowest common parent of the corresponding connected CAEX objects.
- Instance-instance-relations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

  NOTE 1 The AML standard interface class library is specified in 6.3. The interface classes define the semantic of the interface and thus the semantic of the link. A link between interfaces without a reference to an interface class has no semantics.
- COLLADA documents may be interlinked. The corresponding COLLADA interfaces may be any items that have a valid URI. If those nodes are required to be interlinked in CAEX, they shall be published in CAEX by adding a CAEX ExternalInterface to the corresponding object. This ExternalInterface shall be derived from the AML standard interface class “COLLADAInterface” or one of its derivates.

  NOTE 2 The standard interface class “COLLADAInterface” is specified in 6.3.7. Details are intended to be specified in IEC 62714-3.
- PLCopen XML documents may be interlinked by utilizing corresponding PLCopen XML interfaces. If PLCopen XML items are required to be interlinked in CAEX, they shall be published by adding a CAEX ExternalInterface to the corresponding object. This ExternalInterface shall be derived from the AML standard interface class “PLCopenXMLInterface” or one of its derivates.

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

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

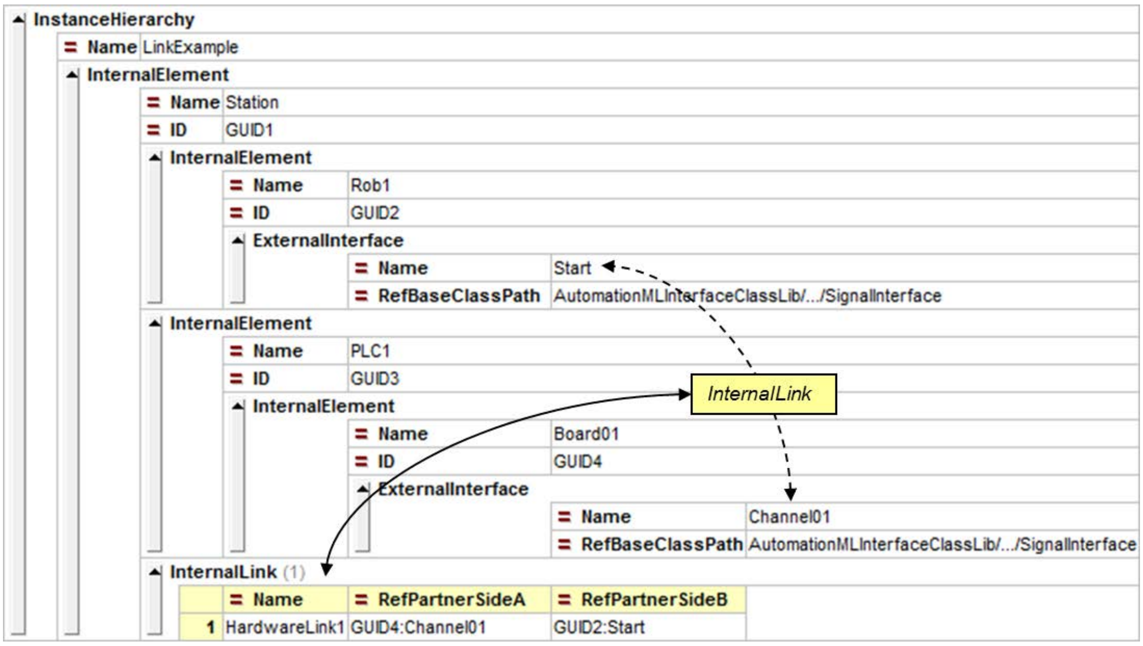


a) Relation as a block diagram b) Relation as an object tree

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

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

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




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

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

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

5.7 AML document reference specification

5.7.1 General

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

5.7.2 Referencing COLLADA documents

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

5.7.3 Referencing PLCopen XML documents

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

5.7.4 Referencing additional documents

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

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

6 AML base libraries

6.1 General

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

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

6.2 General provisions

Regarding AML base libraries, the following provisions apply:

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

6.3 AML interface class library – AutomationMLInterfaceClassLib

6.3.1 General

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

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

Table 3 – Interface classes of the AutomationMLInterfaceClassLib

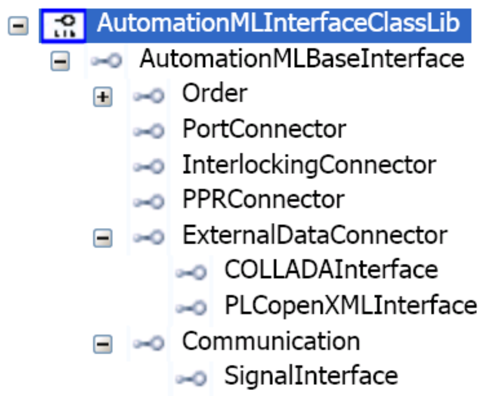
AML InterfaceClass library	InterfaceClass	Description
	AutomationMLBaseInterface	Abstract interface type
	Order	Interface for describing orders
	PortConnector	Interface for describing ports
	PPRConnector	Connector for interlinking products, resources or processes
	ExternalDataConnector	Generic connector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

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

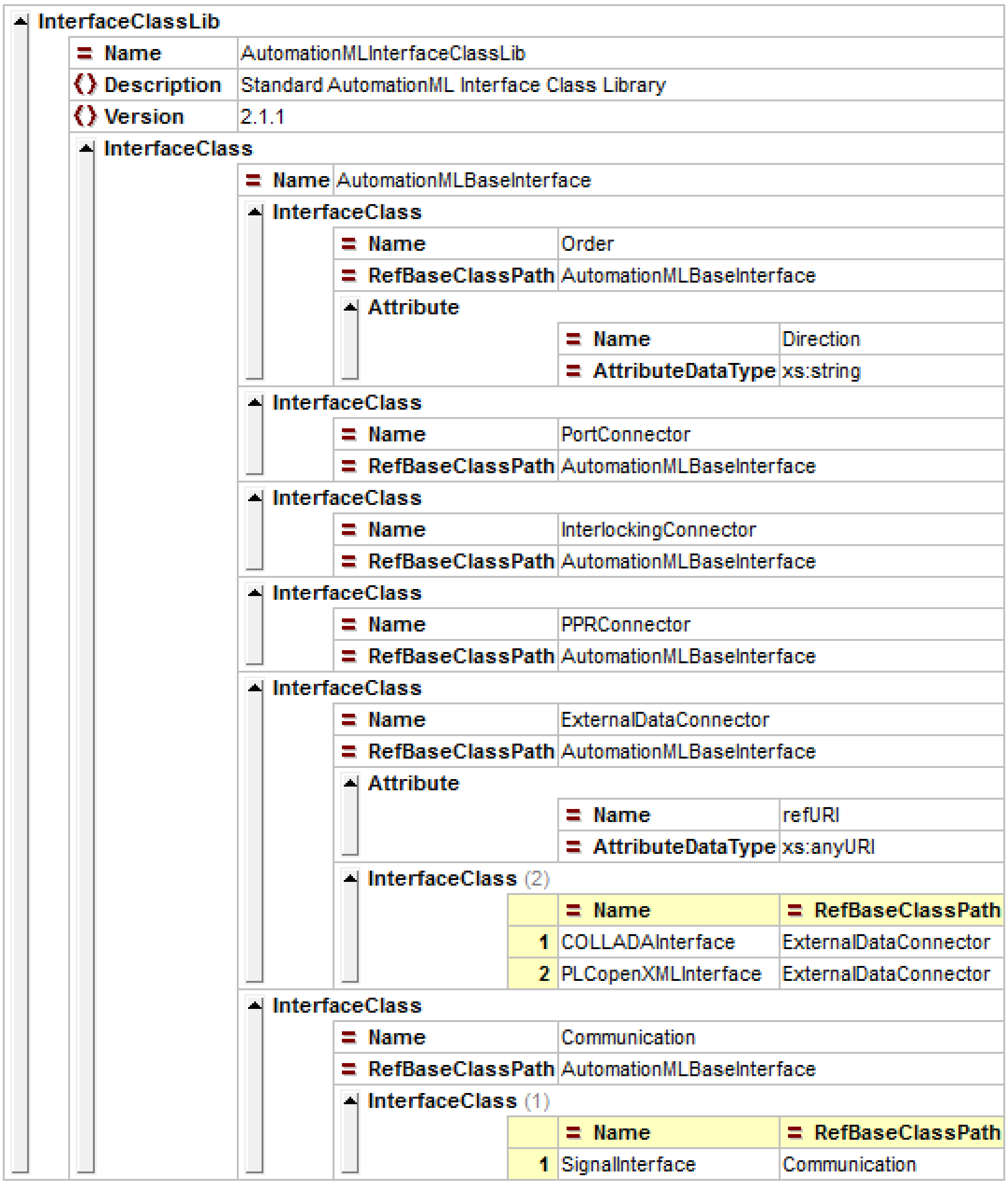


Figure 12 – AML basic interface class library

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

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

6.3.2 InterfaceClass AutomationMLBaseInterface

Table 4 specifies the interface class “AutomationMLBaseInterface”.

Table 4 – InterfaceClass AutomationMLBaseInterface

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

6.3.3 InterfaceClass Order

Table 5 specifies the interface class “Order”.

Table 5 – InterfaceClass Order

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

6.3.4 InterfaceClass PortConnector

Table 6 specifies the interface class “PortConnector”.

Table 6 – InterfaceClass PortConnector

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

6.3.5 InterfaceClass PPRConnector

Table 7 specifies the interface class “PPRConnector”.

Table 7 – InterfaceClass PPRConnector

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

6.3.6 InterfaceClass ExternalDataConnector

Table 8 specifies the interface class “ExternalDataConnector”.

Table 8 – InterfaceClass ExternalDataConnector

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

6.3.7 InterfaceClass COLLADAInterface

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

Table 9 – InterfaceClass COLLADAInterface

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

6.3.8 InterfaceClass PLCopenXMLInterface

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

Table 10 – InterfaceClass PLCopenXMLInterface

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

6.3.9 InterfaceClass Communication

Table 11 specifies the interface class "Communication".

Table 11 – InterfaceClass Communication

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

6.3.10 InterfaceClass SignalInterface

Table 12 specifies the interface class “SignalInterface”.

Table 12 – InterfaceClass SignalInterface

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

6.4 AML basic role class library – AutomationMLBaseRoleClassLib

6.4.1 General

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

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

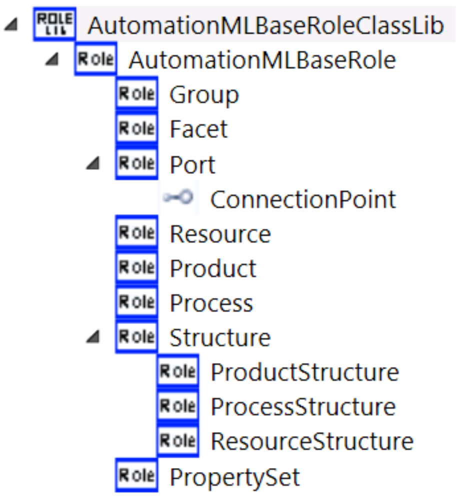


Figure 14 – AML basic role class library

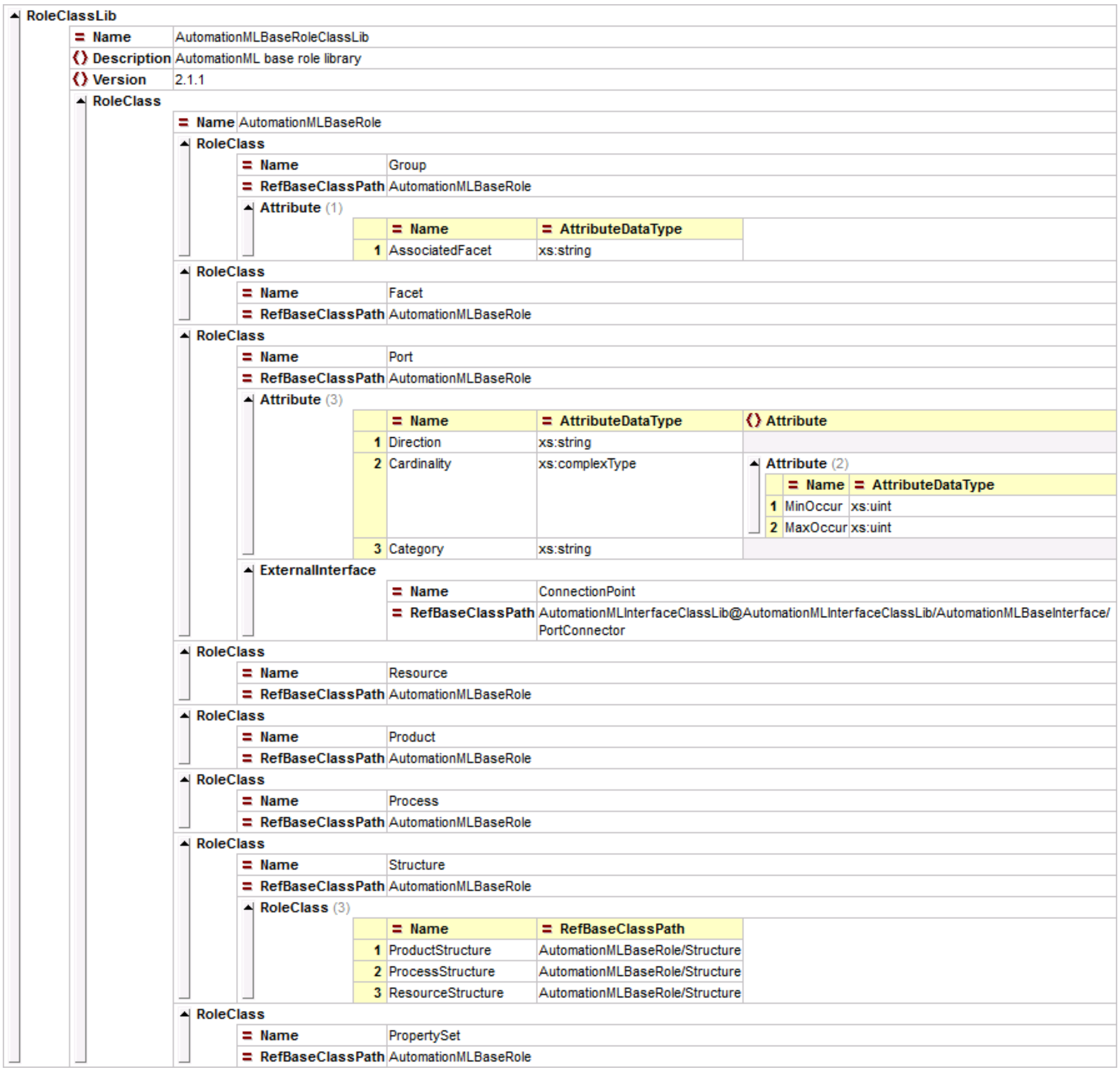


Figure 15 – AutomationMLBaseRoleClassLib

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

Figure 16 – XML text of the AutomationMLBaseRoleClassLib

6.4.2 RoleClass AutomationMLBaseRole

Table 13 specifies the role class “AutomationMLBaseRole”.

Table 13 – RoleClass AutomationMLBaseRole

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

6.4.3 RoleClass Group

Table 14 specifies the role class “Group”.

Table 14 – RoleClass Group

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

6.4.4 RoleClass Facet

Table 15 specifies the role class “Facet”.

Table 15 – RoleClass Facet

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

6.4.5 RoleClass Port

Table 16 specifies the role class “Port”.

Table 16 – Optional attributes for AML Port objects

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

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

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

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

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

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

Table 18 – Interface of the AML Port class

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

6.4.6 RoleClass Resource

Table 19 specifies the role class “Resource”.

Table 19 – RoleClass Resource

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

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

6.4.7 RoleClass Product

Table 20 specifies the role class “Product”.

Table 20 – RoleClass Product

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

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

6.4.8 RoleClass Process

Table 21 specifies the role class “Process”.

Table 21 – RoleClass Process

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

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

6.4.9 RoleClass Structure

Table 22 specifies the role class “Structure”.

Table 22 – RoleClass Structure

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

6.4.10 RoleClass ProductStructure

Table 23 specifies the role class “ProductStructure”.

Table 23 – RoleClass ProductStructure

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

6.4.11 RoleClass ProcessStructure

Table 24 specifies the role class “ProcessStructure”.

Table 24 – RoleClass ProcessStructure

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

6.4.12 RoleClass ResourceStructure

Table 25 specifies the role class “ResourceStructure”.

Table 25 – RoleClass ResourceStructure

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

6.4.13 RoleClass PropertySet

Table 26 specifies the role class “PropertySet”.

Table 26 – RoleClass PropertySet

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

7 Modelling of user-defined data

7.1 General

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

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

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

7.2 User-defined attributes

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

Regarding user-defined attributes, the following provisions apply:

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

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

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

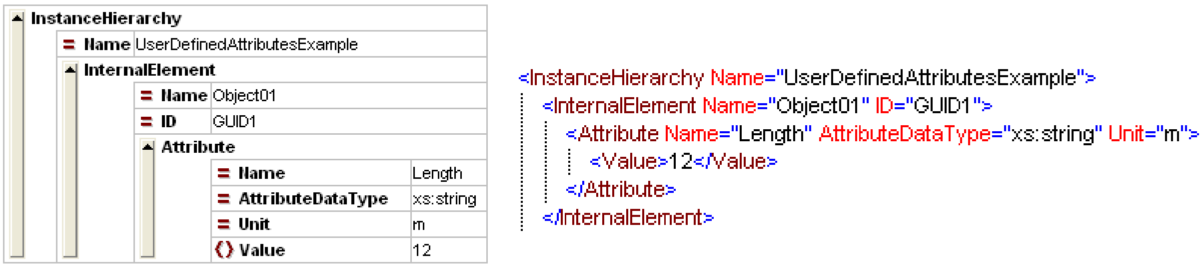


Figure 17 – Example of a user-defined attribute

7.3 User-defined InterfaceClasses

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

Regarding user-defined InterfaceClasses, the following provisions apply:

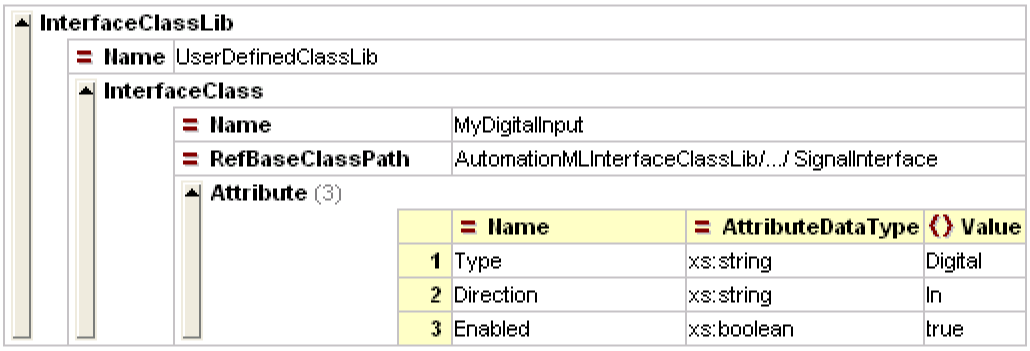
- All user-defined InterfaceClasses shall be stored according to the CAEX InterfaceClass definition in IEC 62424:2008, A.2.5.

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

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

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

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




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

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

7.4 User-defined RoleClasses

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

Regarding user-defined RoleClasses, the following provisions apply:

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

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

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

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

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

RoleClassLib			
	Name	UserDefinedRoleClassLib	
	RoleClass		
		Name	Fence
		RefBaseClassPath	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource

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

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

7.5 User-defined SystemUnitClasses

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

Regarding user-defined SystemUnitClasses, the following provisions apply:

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

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

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

SystemUnitClassLib				
	Name	UserDefinedSystemUnitClassLib		
	SystemUnitClass			
		Name	Robot1234	
		SupportedRoleClass		
			RefRoleClassPath	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource
	SystemUnitClass			
		Name	SpecialRobot1234	
		RefBaseClassPath	UserDefinedSystemUnitClassLib/Robot1234	

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

Figure 20 – Examples for different user-defined SystemUnitClasses

7.6 User-defined InstanceHierarchies

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

Regarding user-defined InstanceHierarchies, the following provisions apply:

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

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

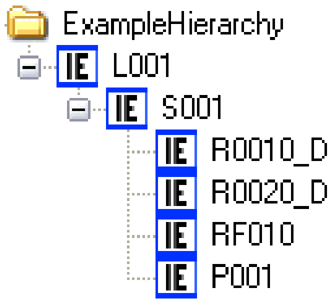


Figure 21 – Example of a user-defined InstanceHierarchy

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

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

Figure 22 – AML representation of a user-defined InstanceHierarchy

8 Extended AML concepts

8.1 General overview

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

8.2 AML Port object

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

Regarding AML Ports, the following provisions apply:

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

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

8.3 AML Facet object

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

Regarding AML Facets, the following provisions apply:

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

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

8.4 AML Group object

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

Regarding AML Group objects, the following provisions apply:

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

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

  NOTE 1 Thus, Group objects can be nested.

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

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

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

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

8.5 AML PropertySet

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

Regarding the PropertySet concept, the following provisions apply:

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

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

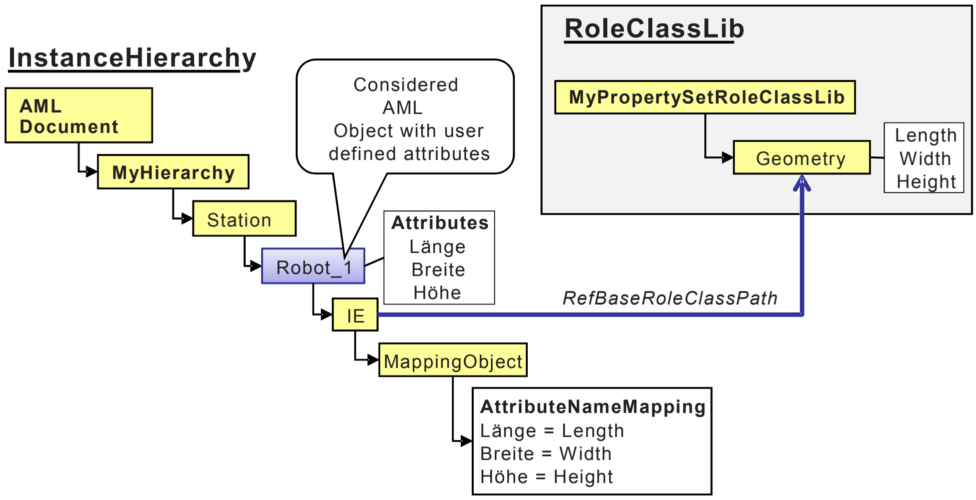
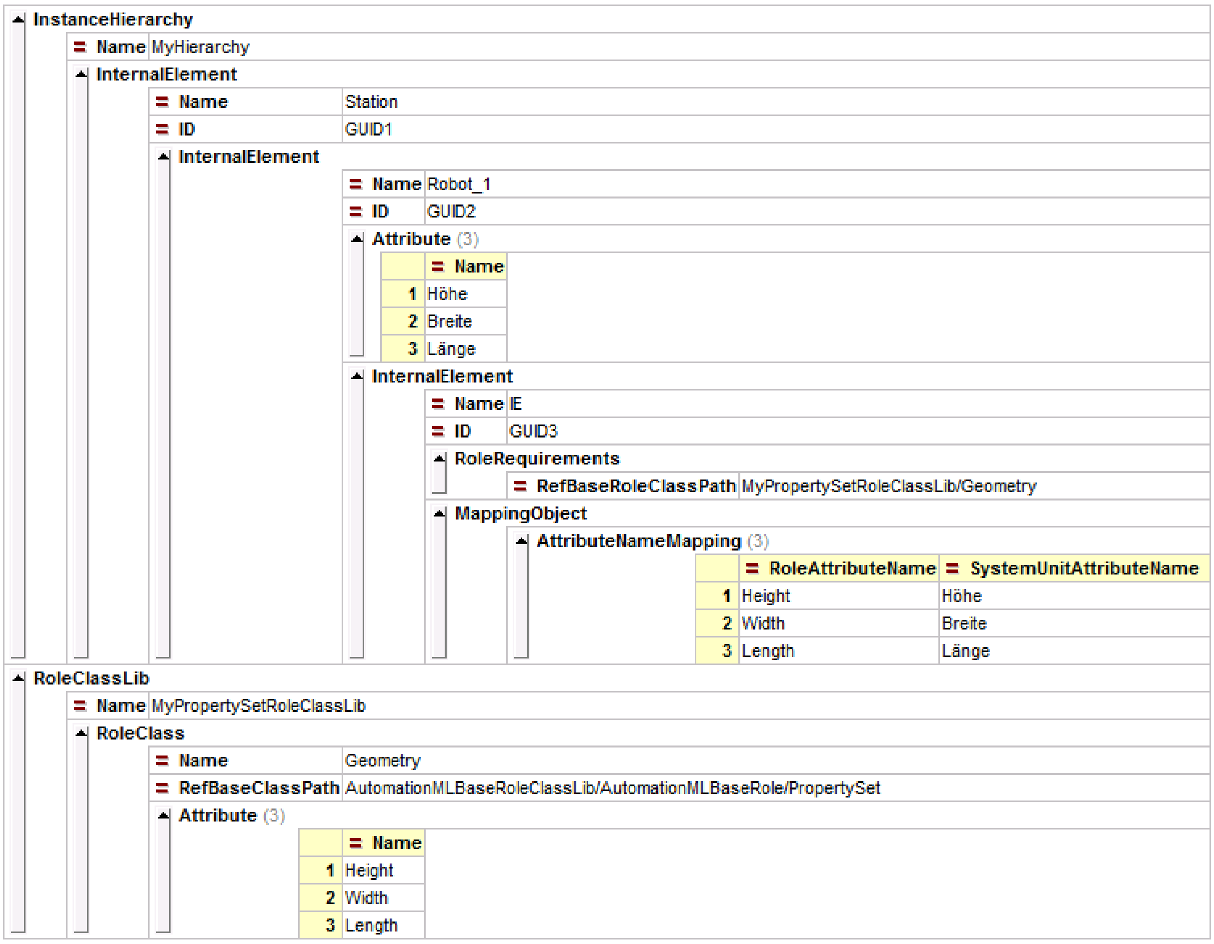


Figure 23 – Example illustrating the PropertySet concept




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

Figure 24 – XML text of the PropertySet example

8.6 Support of multiple roles

In addition to IEC 62424:2008, A.3.18, this part of IEC 62714 defines how to specify multiple roles support for an object instance. Multiple roles are of interest, if an object can have multiple functionalities. An example is a device that is a scanner, a printer or a fax device at the same time. Subclause A.2.7 gives an overview and describes a corresponding example.

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

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

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

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

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

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

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

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

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

8.7 Splitting of AML top-level data into different documents

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

8.8 Internationalization

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

8.9 Version information of AML objects

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

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

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

Annex A
(informative)

General introduction into the Automation Markup Language

A.1 General Automation Markup Language concepts

A.1.1 The Automation Markup Language architecture

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

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

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

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

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

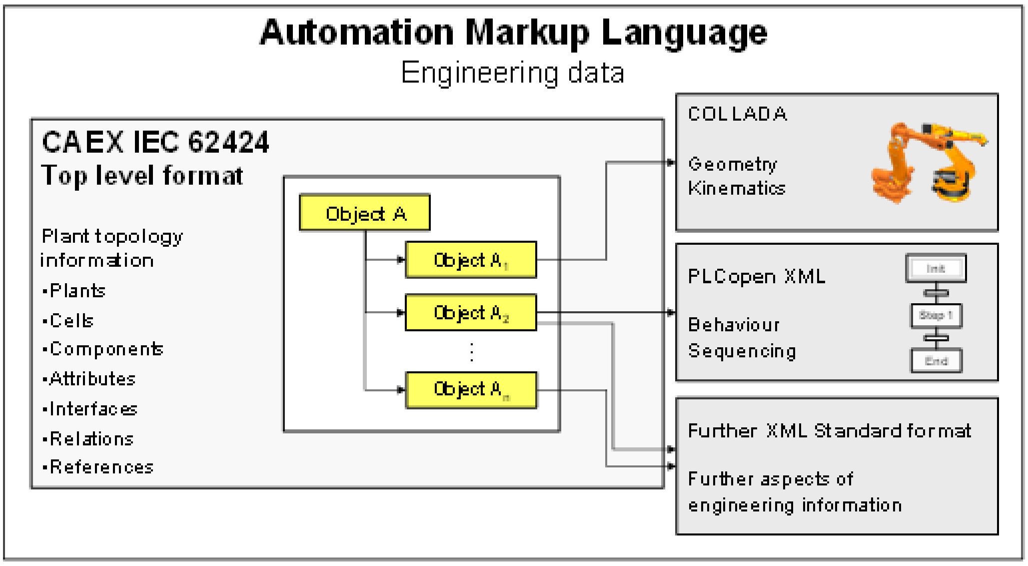


Figure A.1 – AML general architecture

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

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

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

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

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

A.1.2 Modelling of plant topology information

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

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

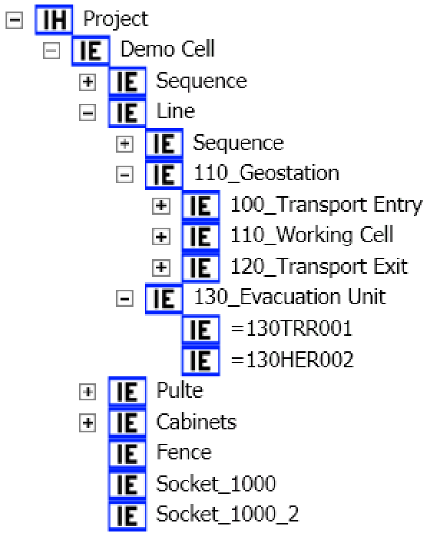


Figure A.2 – Plant topology with AML

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

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

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

A.1.3 Referencing geometry and kinematics information

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

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

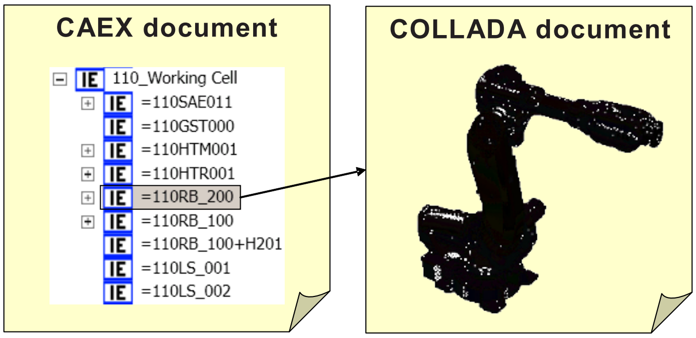


Figure A.3 – Reference from CAEX to a COLLADA document

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

A.1.4 Referencing logic information

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

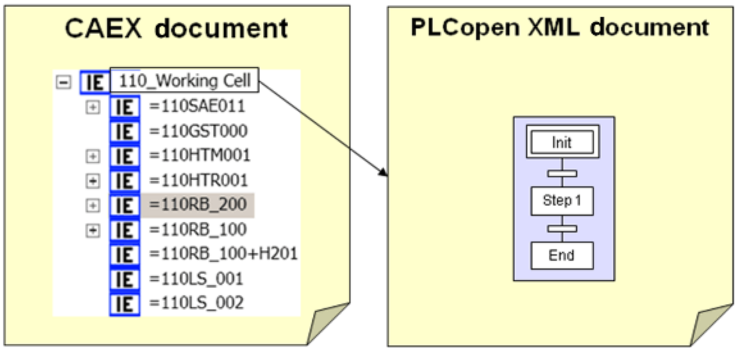


Figure A.4 – Reference from a CAEX to a PLCopen XML document

A.1.5 Modelling of relations

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

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

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

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

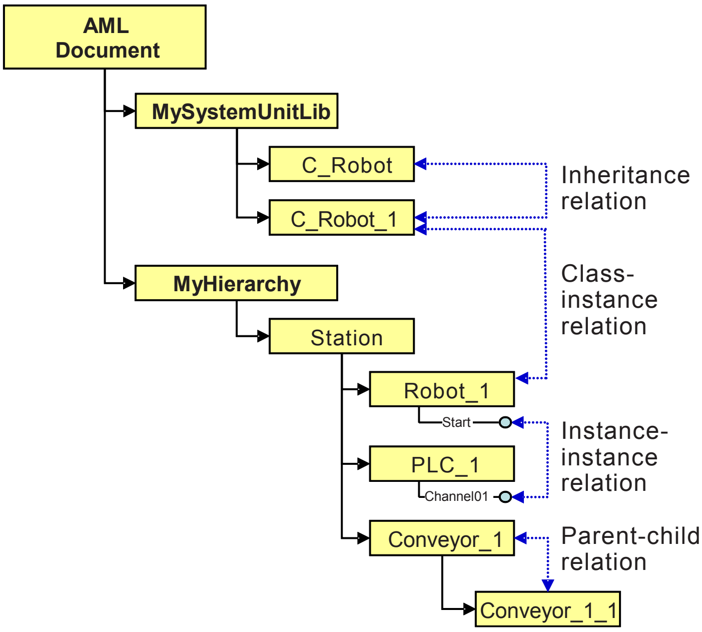


Figure A.5 – Relations in AML

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

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

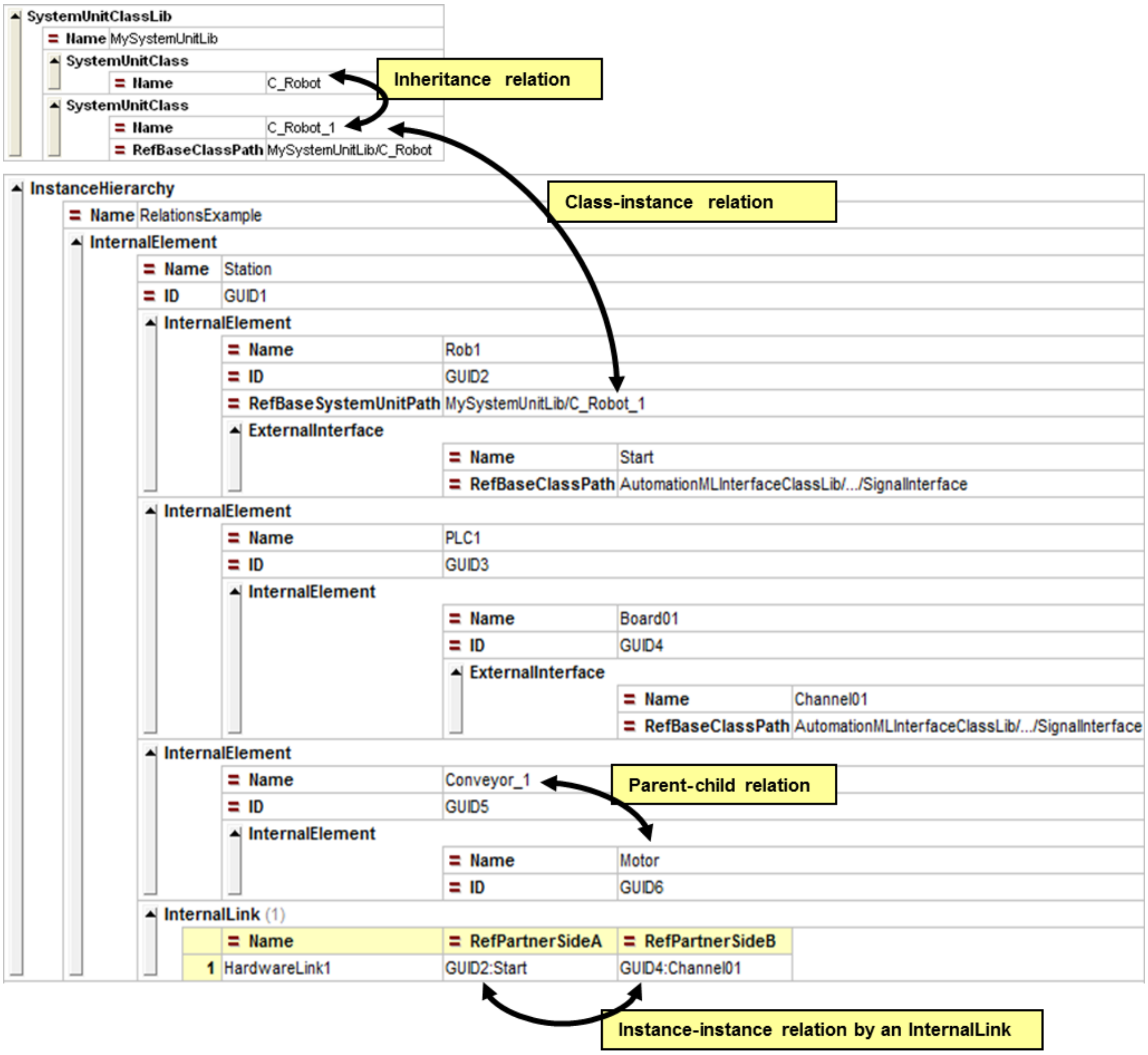


Figure A.6 – XML description of the relations example

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

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

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

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

A.2 Extended AML concepts and examples

A.2.1 General overview

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

Table A.1 – Overview of major extended AML concepts

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

A.2.2 AML Port concept

A.2.2.1 Concept description

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

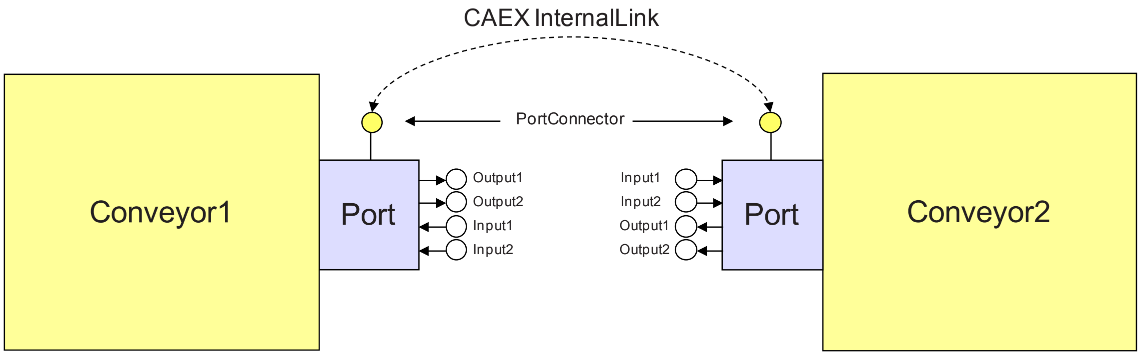


Figure A.9 – Port concept

A.2.2.2 Example

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

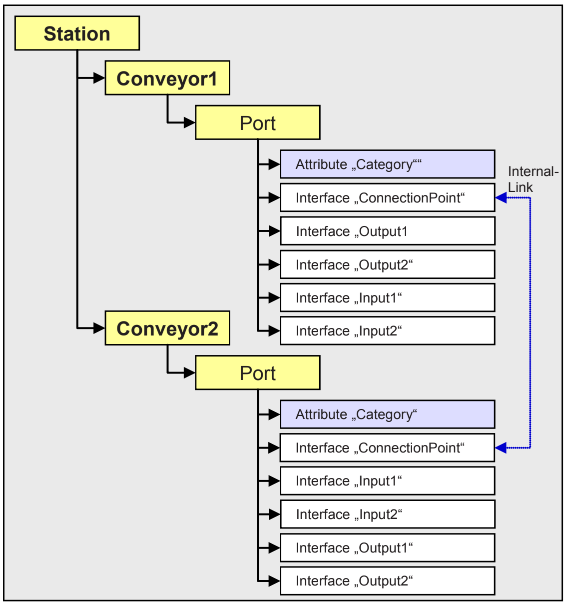


Figure A.10 – Example describing the AML Port concept

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

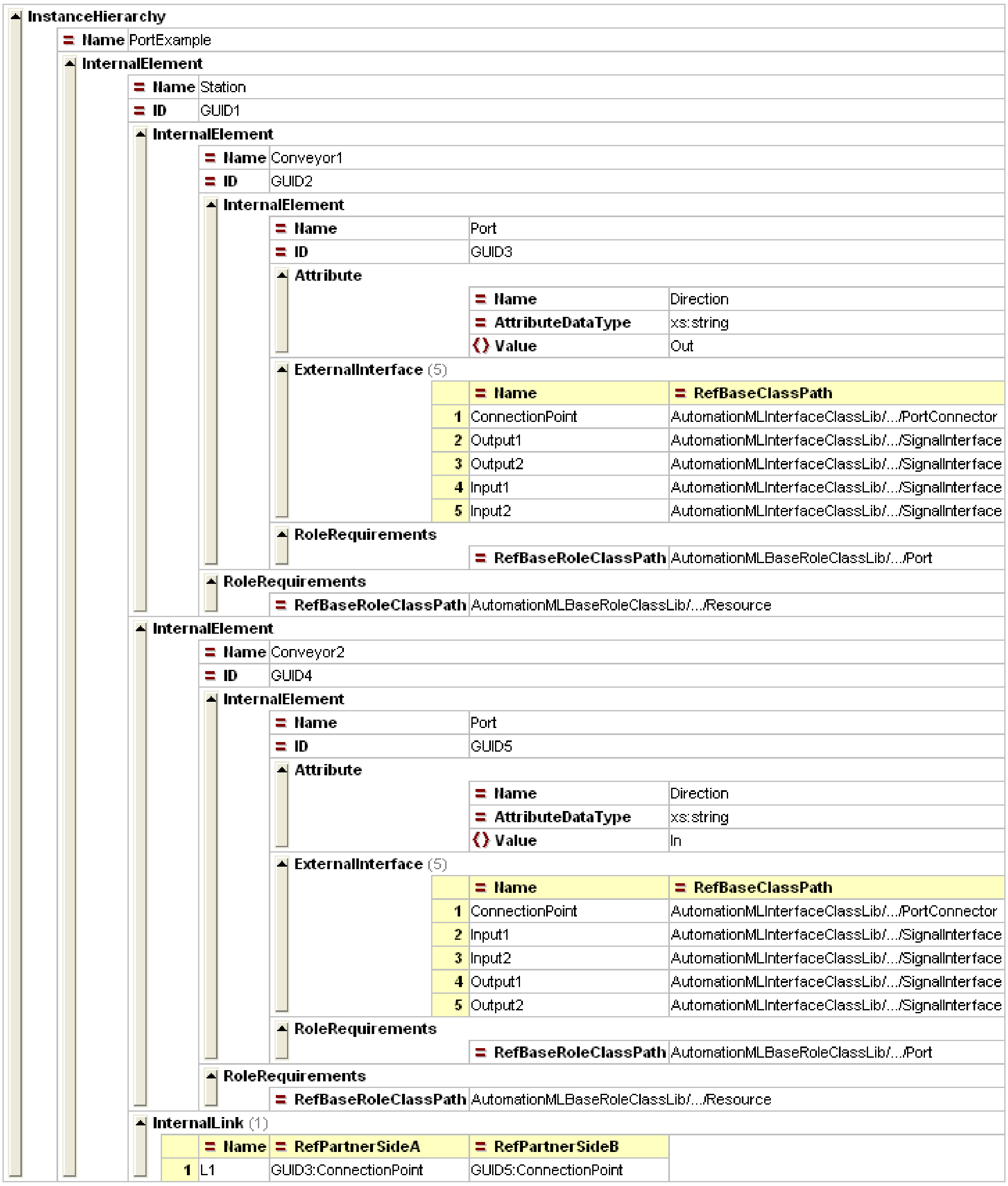


Figure A.11 – XML description of the AML Port concept

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

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

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

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

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

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

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

A.2.3 AML Facet concept

A.2.3.1 Concept description

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

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

A.2.3.2 Example

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

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

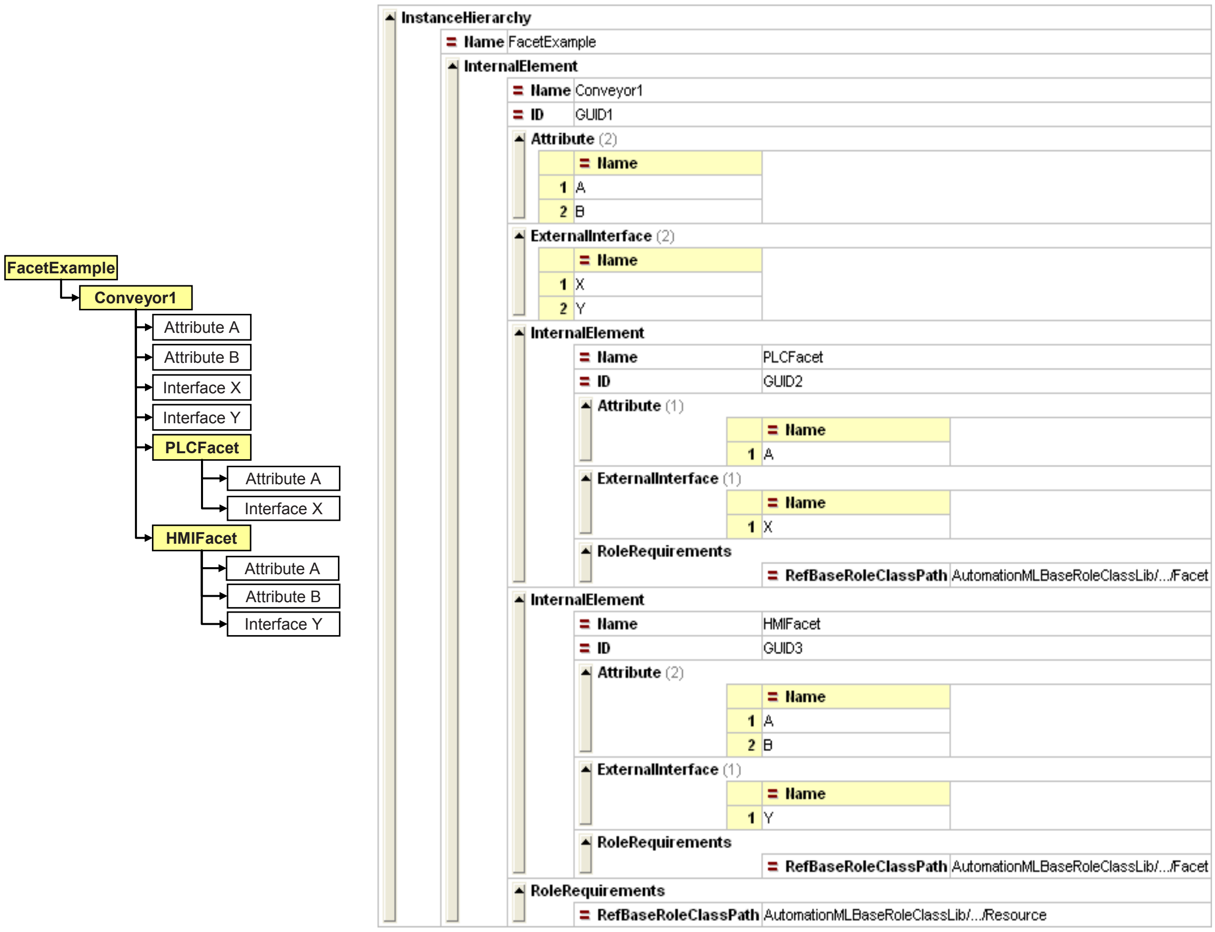


Figure A.14 – AML Facet example

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

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

A.2.4 AML Group concept

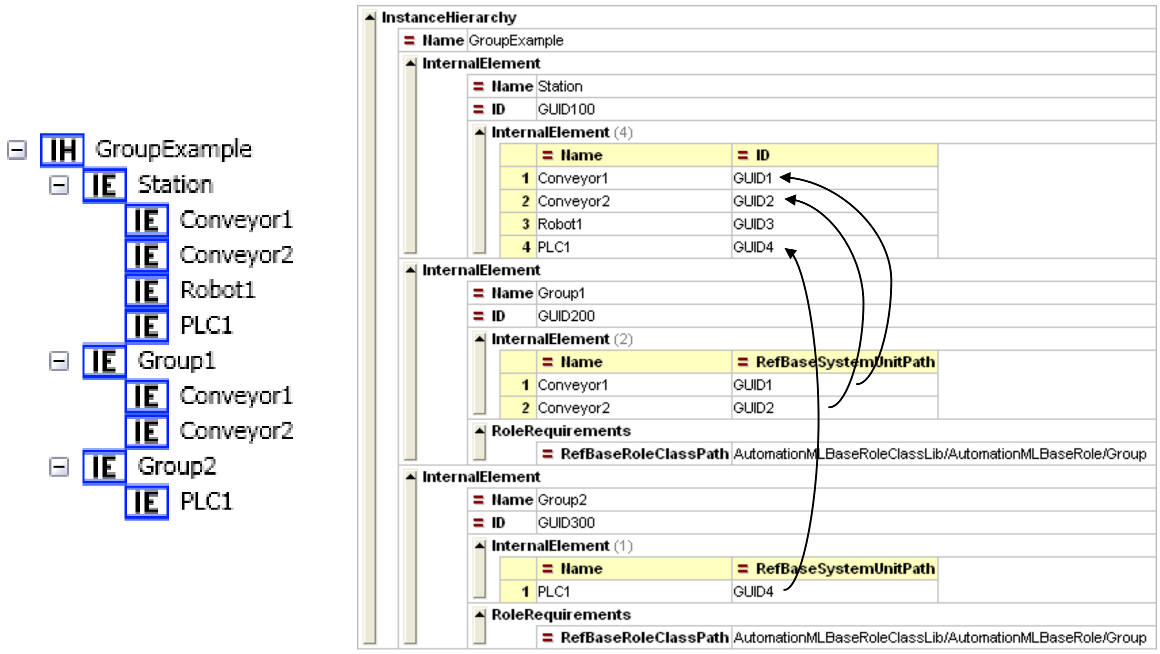
A.2.4.1 Concept description

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

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

A.2.4.2 Example

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



a) Hierarchy b) Grid view

Figure A.16 – AML Group example

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


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

A.2.4.3 Combination of the Group and Facet concept

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

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

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

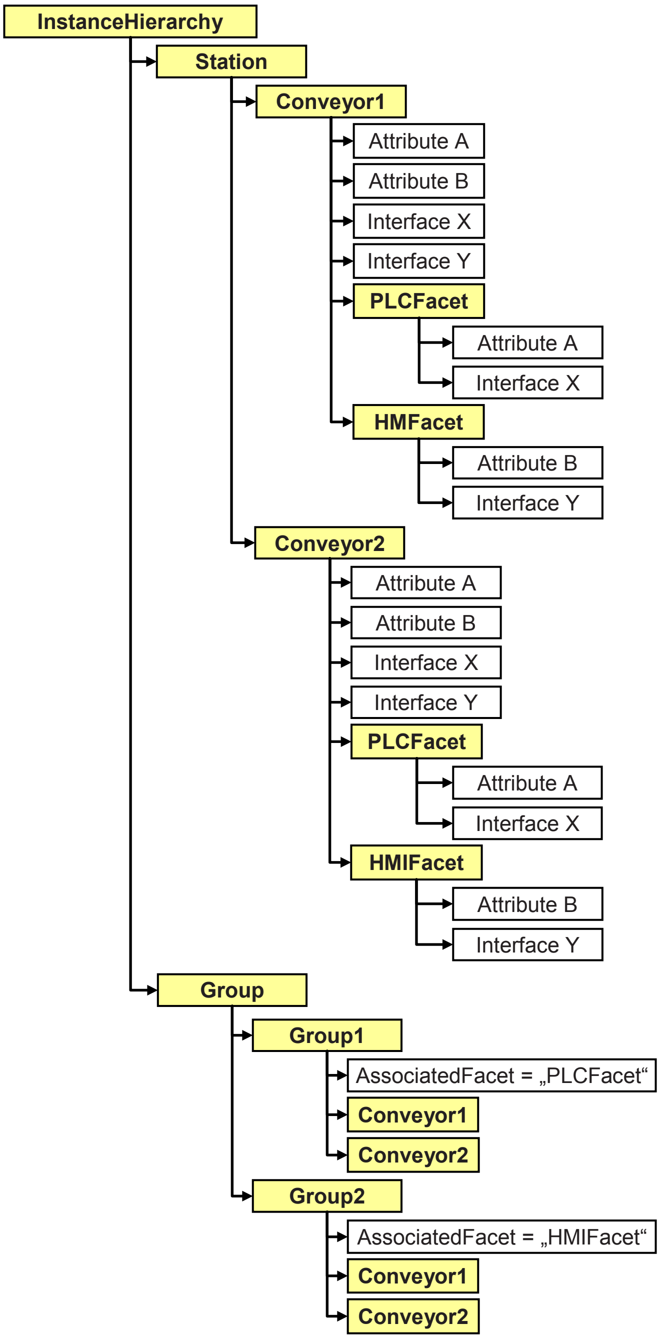


Figure A.18 – Combination of the Facet and Group concept

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

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

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

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

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

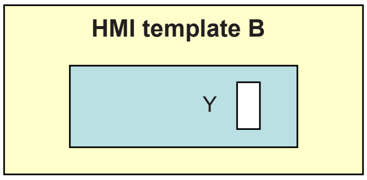


Figure A.20 – Generic HMI template "B" visualizing a process variable "Y" of a conveyor

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

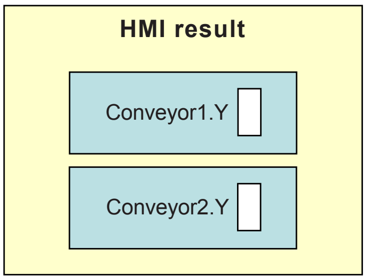


Figure A.21 – Generated HMI result "B" visualizing both conveyors with individual process variables

A.2.5 PropertySet concept

A.2.5.1 Concept description

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

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

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

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

A.2.5.2 Example

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

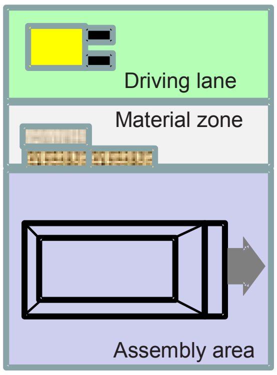


Figure A.22 – PropertySet example

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

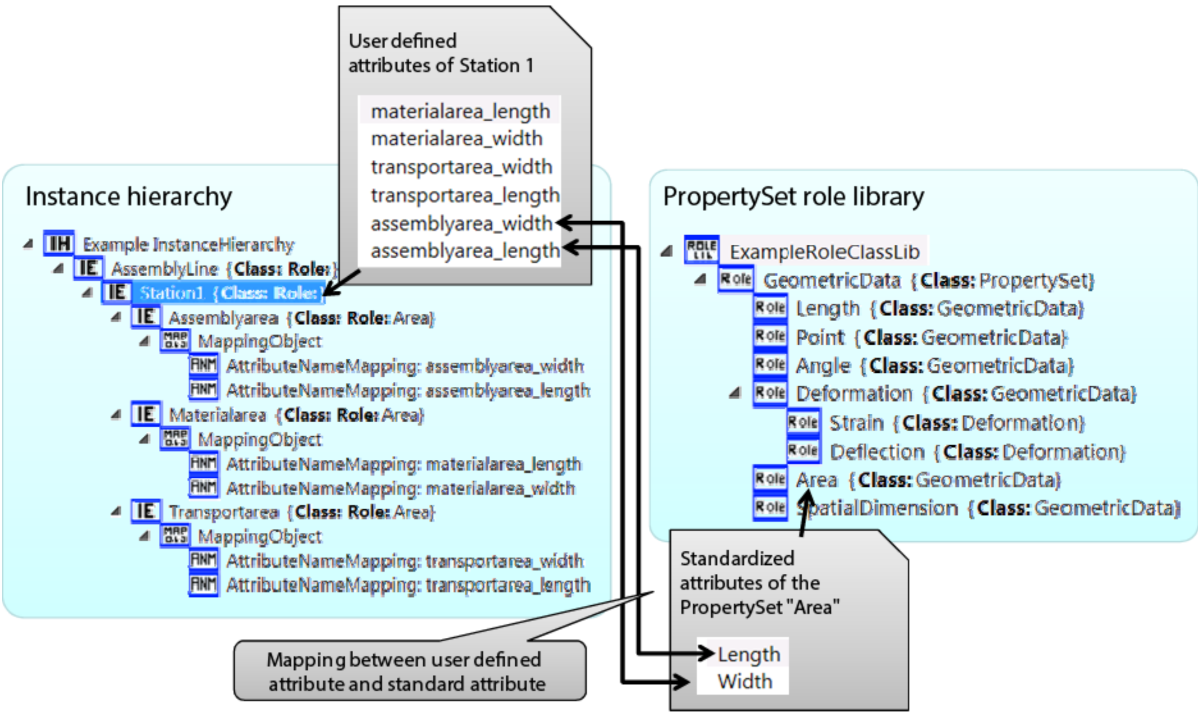


Figure A.23 – PropertySet example

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

Figure A.24 – XML text for the instance hierarchy

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

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

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

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

A.2.6 Process-Product-Resource concept

A.2.6.1 Concept description

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

- In a resource centric view, resources form the central component within the model: they execute processes and handle products. In AML, a resource is an entity involved in production including plants, robots, machines, their state, equipment, possible messages and so on. According to this, resources can be hardware components of a production system, as well as software systems, e.g. SCADA systems. Within AML, resources are typically modelled in a plant hierarchy forming the plant topology.

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

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

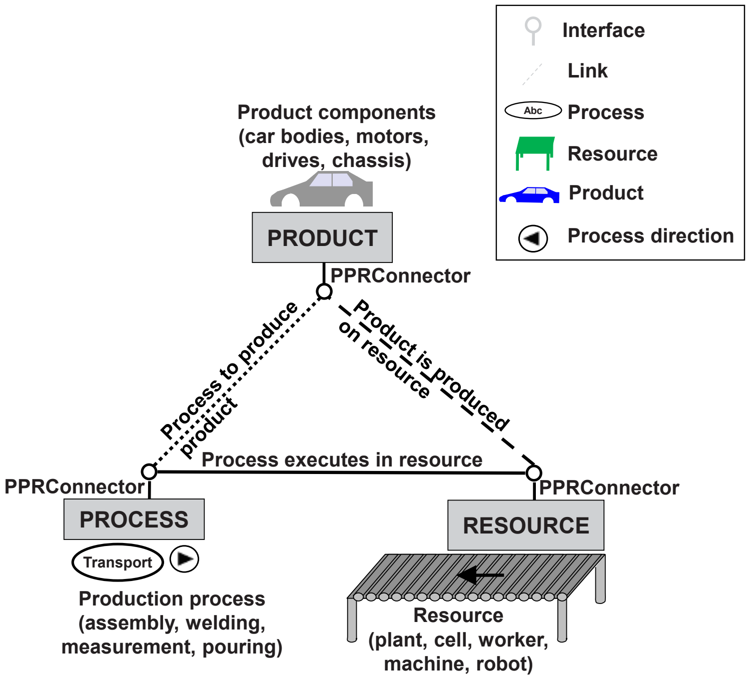


Figure A.26 – Base elements of the Product-Process-Resource concept

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

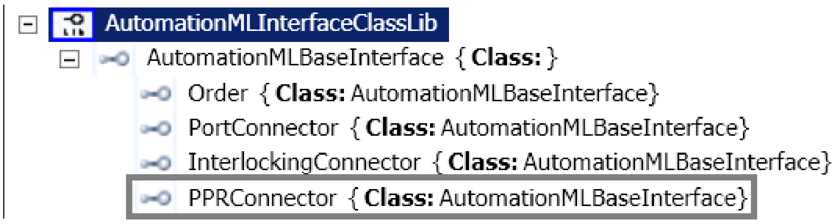


Figure A.27 – PPRConnector interface

A.2.6.2 Example

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

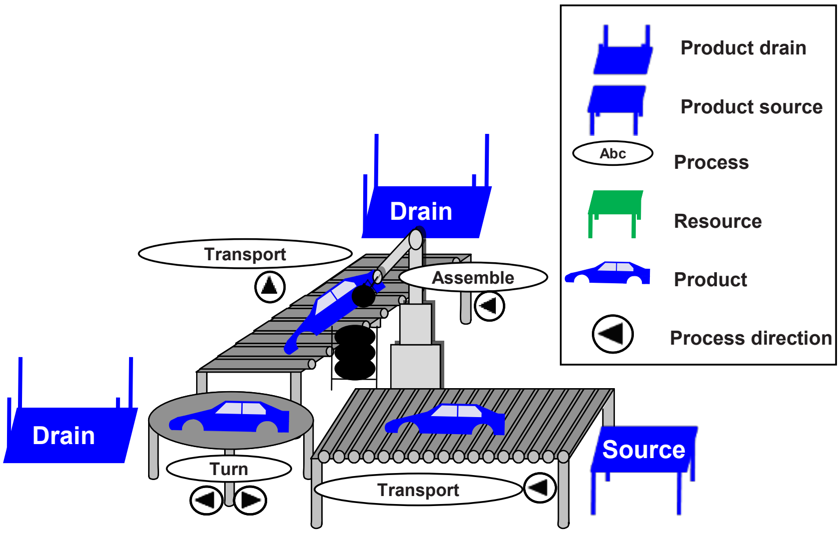


Figure A.28 – Example for the Product-Process-Resource concept

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

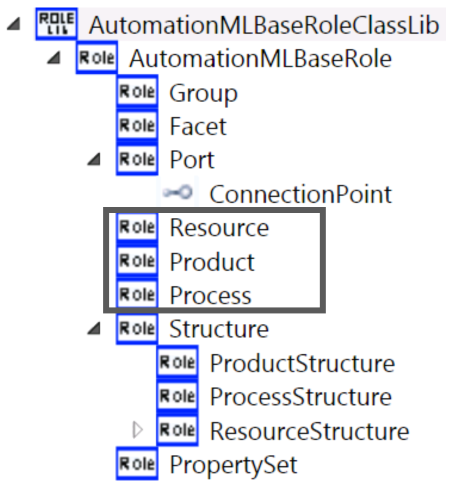


Figure A.29 – AML roles required for the Process-Product-Resource concept

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

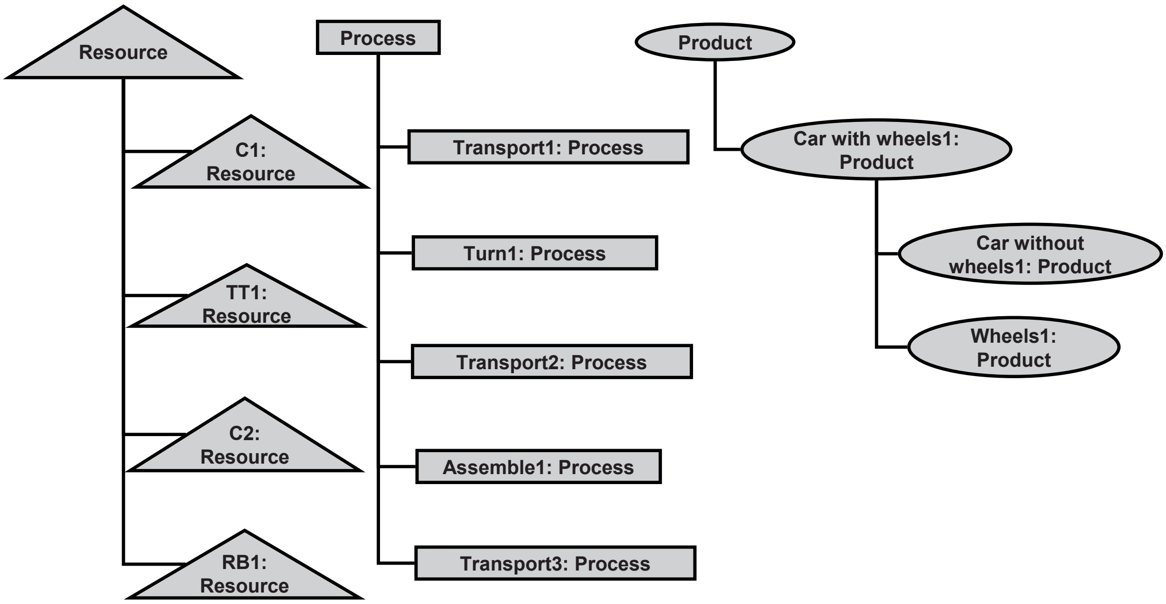


Figure A.30 – Elements of the example

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

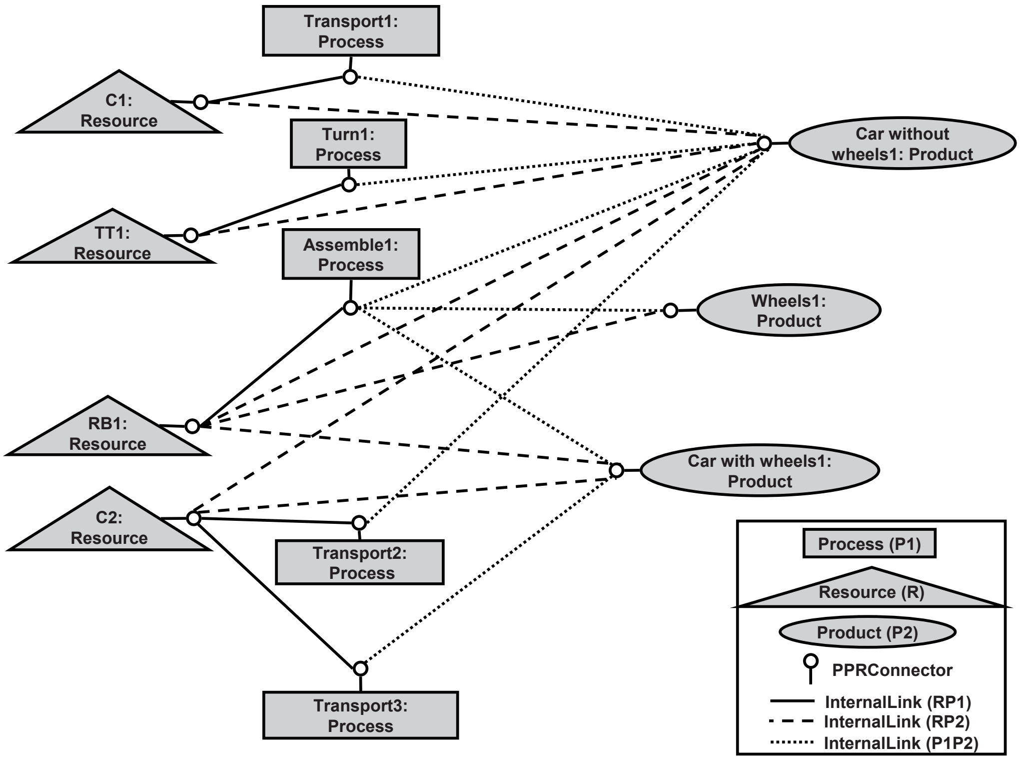


Figure A.31 – Links within the example

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

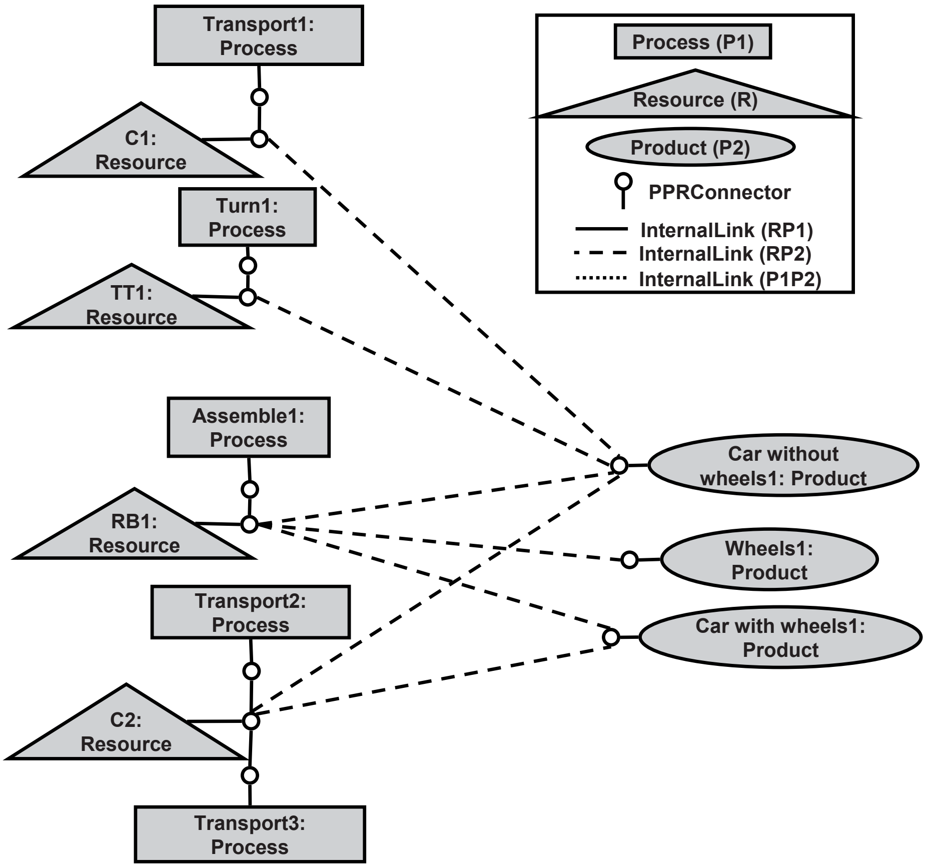


Figure A.32 – Links of the resource centric view on the example

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

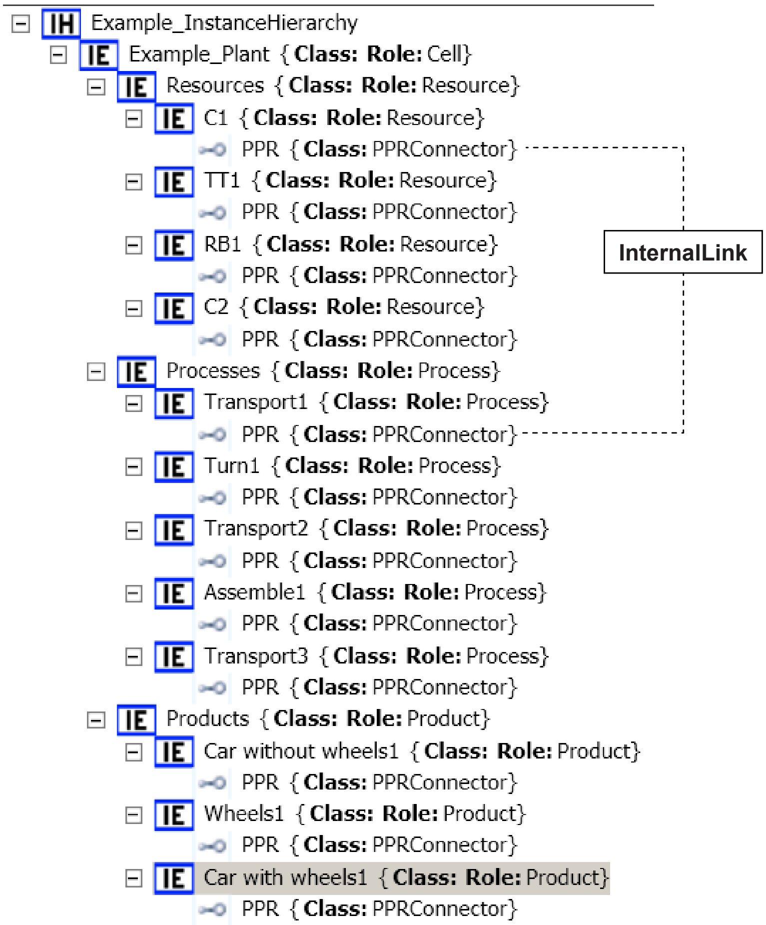


Figure A.33 – InstanceHierarchy of the example in AML

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

InternalElement

Name	Example_Plant
ID	GUID1

InternalElement (3)

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

Resources – InternalElement (4)

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

Processes – InternalElement (5)

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

Products – InternalElement (3)

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

Figure A.34 – InternalElements of the example

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

InternalLink (12)

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

Figure A.35 – InternalLinks of the example

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

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

Figure A.36 – InstanceHierarchy of the example in XML

A.2.7 Support of multiple roles

In addition to IEC 62424:2008, A.2.9, AML defines how to specify multiple role support for an object instance. Multiple roles are of interest, if an object can have multiple functionalities. An example is a multi-functional device that is a scanner, a printer or a fax device at the same time. Provisions regarding multiple roles are specified in 8.5.

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

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

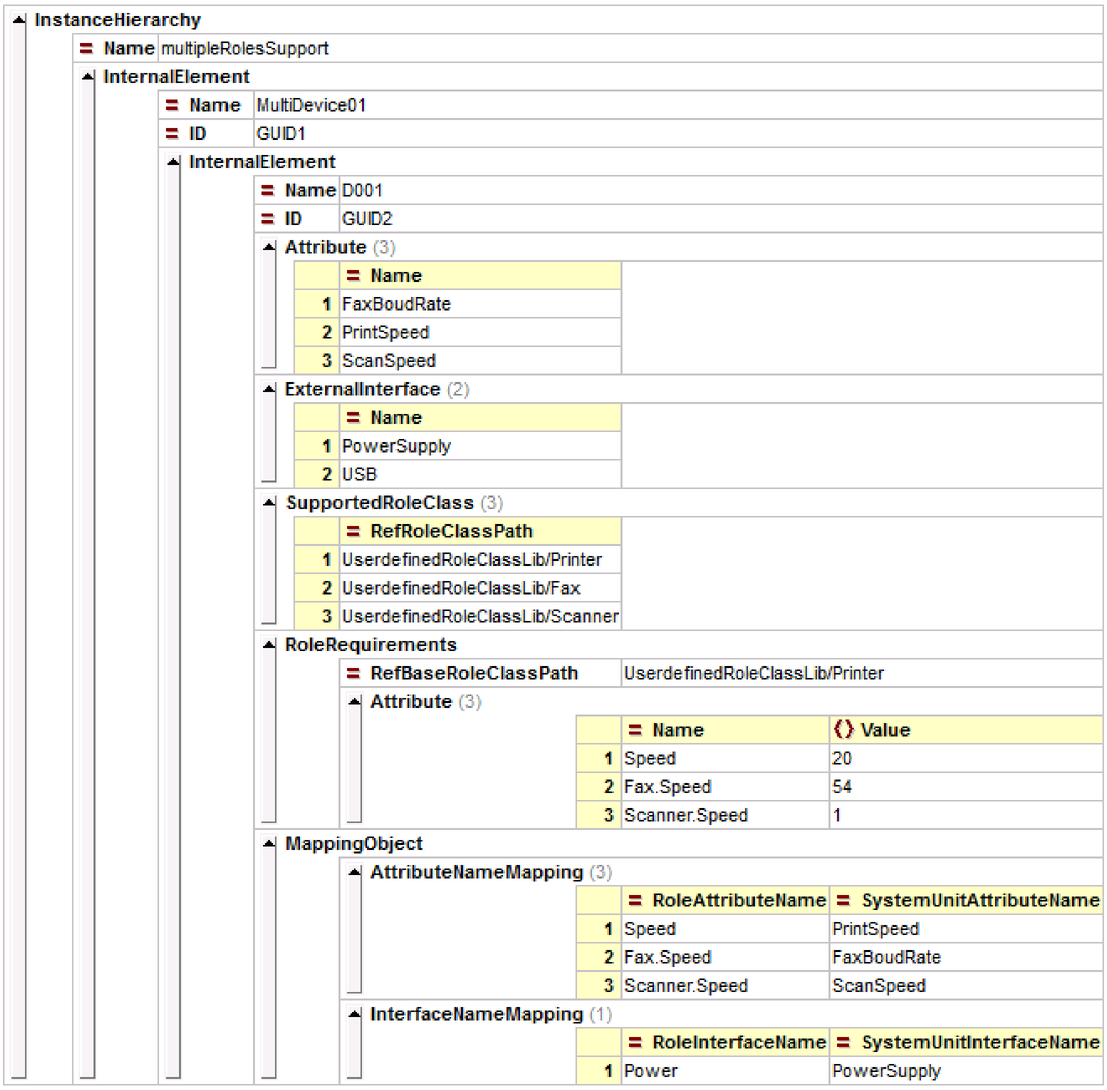


Figure A.37 – Example of a user-defined instance supporting multiple roles

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

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

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

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

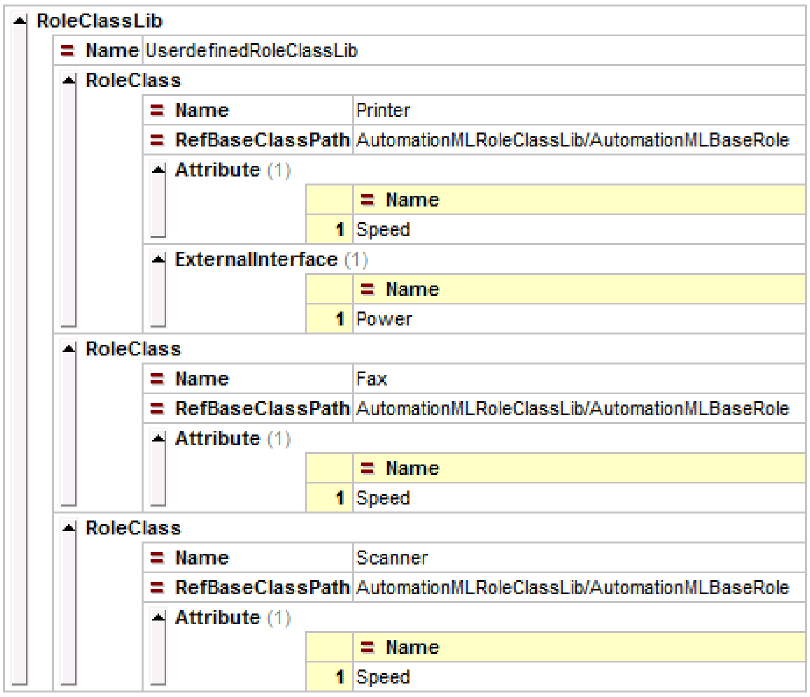


Figure A.39 – AML Role class library corresponding to the multiple role definition example

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


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

Annex B
(informative)

XML Representation of AML Libraries

B.1 AutomationMLBaseRoleClassLib


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

B.2 AutomationMLInterfaceClassLib


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

Bibliography

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

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

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

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

___

2 To be published.

This page deliberately left blank

British Standards Institution (BSI)

BSI is the national body responsible for preparing British Standards and other standards-related publications, information and services.

BSI is incorporated by Royal Charter. British Standards and other standardization products are published by BSI Standards Limited.

About us

We bring together business, industry, government, consumers, innovators and others to shape their combined experience and expertise into standards -based solutions.

The knowledge embodied in our standards has been carefully assembled in a dependable format and refined through our open consultation process. Organizations of all sizes and across all sectors choose standards to help them achieve their goals.

Information on standards

We can provide you with the knowledge that your organization needs to succeed. Find out more about British Standards by visiting our website at bsigroup.com/standards or contacting our Customer Services team or Knowledge Centre.

Buying standards

You can buy and download PDF versions of BSI publications, including British and adopted European and international standards, through our website at bsigroup.com/shop, where hard copies can also be purchased.

If you need international and foreign standards from other Standards Development Organizations, hard copies can be ordered from our Customer Services team.

Subscriptions

Our range of subscription services are designed to make using standards easier for you. For further information on our subscription products go to bsigroup.com/subscriptions.

With **British Standards Online (BSOL)** you'll have instant access to over 55,000 British and adopted European and international standards from your desktop. It's available 24/7 and is refreshed daily so you'll always be up to date.

You can keep in touch with standards developments and receive substantial discounts on the purchase price of standards, both in single copy and subscription format, by becoming a **BSI Subscribing Member**.

PLUS is an updating service exclusive to BSI Subscribing Members. You will automatically receive the latest hard copy of your standards when they're revised or replaced.

To find out more about becoming a BSI Subscribing Member and the benefits of membership, please visit bsigroup.com/shop.

With a **Multi-User Network Licence (MUNL)** you are able to host standards publications on your intranet. Licences can cover as few or as many users as you wish. With updates supplied as soon as they're available, you can be sure your documentation is current. For further information, email bsmusales@bsigroup.com.

Revisions

Our British Standards and other publications are updated by amendment or revision.

We continually improve the quality of our products and services to benefit your business. If you find an inaccuracy or ambiguity within a British Standard or other BSI publication please inform the Knowledge Centre.

Copyright

Useful Contacts:

Customer Services
Tel: +44 845 086 9001
Email (orders): orders@bsigroup.com
Email (enquiries): cservices@bsigroup.com

Subscriptions
Tel: +44 845 086 9001
Email: subscriptions@bsigroup.com

Knowledge Centre
Tel: +44 20 8996 7004
Email: knowledgecentre@bsigroup.com

Copyright & Licensing
Tel: +44 20 8996 7070
Email: copyright@bsigroup.com

BSI Group Headquarters

389 Chiswick High Road London W4 4AL UK

bsi.

...making excellence a habit.™