

ICS 25.040.01; 35.240.50

Kombinieren von OPC Unified Architecture und Automation Markup Language

Combining OPC Unified Architecture and Automation Markup Language

Combinaison de l'architecture unifiée OPC et l'Automation Markup Language

Zur Erstellung einer DIN SPEC können verschiedene Verfahrensweisen herangezogen werden:
Das vorliegende Dokument wurde nach den Verfahrensregeln einer PAS erstellt.

Gesamtumfang 68 Seiten

Dieses Dokument wurde durch die im Vorwort genannten Verfasser erarbeitet und verabschiedet.

Contents

Page

Foreword	4
1 Scope	6
2 Normative references	6
3 Terms and definitions	7
4 Symbols and abbreviated terms	7
5 AML model mapping to OPC Unified Architecture information model	8
5.1 AML modelling concept	8
5.2 How to map OPC Unified Architecture Namespaces	10
5.3 How to map AML model elements	11
5.4 Mapping of AML XML DataTypes to OPC Unified Architecture DataTypes	18
5.5 AML base types for OPC Unified Architecture information model	20
5.5.1 ObjectTypes	20
5.5.2 ReferenceTypes	23
5.5.3 VariableTypes	25
5.5.4 Structure DataTypes	28
5.5.5 OPC Unified Architecture Objects used to organize the AddressSpace structure	29
5.6 Mapping of AML libraries	32
5.7 Mapping of multiple AML documents	32
6 Integration of OPC Unified Architecture configuration information into AML models	34
7 Relation to other standards and specifications	35
Annex A (informative) Industrial application	37
A.1 General	37
A.2 Use Case 1: Information lifecycle management	40
A.3 Use Case 2: Up-to-date description of the system as-is	41
A.4 Use Case 3: Information exchange (e.g. asset information, quality information, diagnostic data, etc.) with MES or SCADA systems for system operation	42
A.5 Use Case 4: Lossless exchange of OPC Unified Architecture system configuration	43
A.6 Use Case 5: Communicate/Operationalize AML by means of OPC Unified Architecture	43
A.7 Use Case 6: Mixed simulation environments	45
A.8 Use Case 7: Lossless storage and retrieval of system engineering information for system maintenance, repair, overhaul (MRO)	45
A.9 Use Case 8: Manufacturing change management	46
A.10 Use Case 9: Lossless storage and retrieval of system engineering information for manufacturing system reconfiguration	47
Annex B (informative) Mapping example	49
Annex C (informative) AML base types NodeSet	62
Bibliography	67

Foreword

This DIN SPEC has been developed according to the PAS (Publicly Available Specification) procedure. DIN SPECs developed according to the PAS procedure are prepared in workshops that do not necessarily include all stakeholders.

This DIN SPEC (PAS) is the result of the INS (Innovation with Norms and Standards) project "Plug-and-Work – Universelle Schnittstellen für die Automatisierung" funded by the German Federal Ministry for Economic Affairs and Energy (BMWi).

The following initiators and authors developed and approved this document:

- Fraunhofer-Institut IOSB
Dr. Miriam Schleipen
- Fraunhofer-Institut IOSB
Robert Henßen
- Softing Industrial Automation GmbH
Hans Endl
- Enterprise Information Systems (EIS), Rheinische Friedrich-Wilhelms-Universität Bonn
Irlan Grangel-González
- Technische Universität München
Veit Hammerstingl
- Fraunhofer-Institut IPA
Günther Hörcher
- Fraunhofer-Institut IPA
Olha Meyer
- Fraunhofer-Institut IPK
Sebastian Neumeyer
- Otto-v.-Guericke Universität
Arndt Lüder
- Otto-v.-Guericke Universität
Nicole Schmidt
- Fraunhofer IOSB-INA Kompetenzzentrum Industrial
Florian Pethig
- ESR Pollmeier
Stefan Pollmeier
- CAN in Automation (CiA)
Thilo Schumann

- Lehrstuhl für Werkzeugmaschinen Werkzeugmaschinenlabor WZL der RWTH Aachen
Simon Storms
- ProSTEP iViP e.V.
Steven Vettermann
- DFKI GmbH
Stephan Weyer
- TU Dresden
Sachari Wassilew

There are currently no standards covering this subject. DIN SPECs are not part of the body of German standards. No draft of this DIN SPEC has been published.

Despite great effort to ensure that technical and non-technical descriptions are correct, reliable and precise, the workshop can make no express or implied warranty for the correctness of the document. This document is used on the understanding that the workshop shall not be held liable for damage or loss of any kind. Use of this DIN SPEC does not exempt users from being responsible for their own actions and they therefore use this DIN SPEC at their own risk.

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

1 Scope

This specification specifies the combination of AML engineering data with OPC Unified Architecture online information such as process data and diagnostic information.

The AML data format, developed by AutomationML e.V. has been standardized in the IEC 62714 series of standards. This format is an XML-based data exchange format which enables the transfer of engineering data for manufacturing systems within a heterogeneous engineering tool landscape.

The OPC Unified Architecture developed by the OPC Foundation is standardised in IEC 62541. It defines platform independent communication mechanisms for online data exchange and generic, extensible, and object-oriented modelling capabilities for the information a system wants to expose.

The combination of AML and OPC Unified Architecture is divided into two parts:

1. Providing an AML model via OPC Unified Architecture; this involves the AML integration into OPC Unified Architecture. The goal is to communicate, exchange, and operationalize AML by means of OPC Unified Architecture. The purpose is to simplify the creation of OPC Unified Architecture information models based on existing AML data. This can be used for re-engineering and maintenance use cases where the AML model evolves over time.
2. Accessing an OPC Unified Architecture server via AML; this involves the integration of OPC Unified Architecture information into AML. The goal is the lossless exchange of OPC Unified Architecture system configuration within AML models to simplify the setup of OPC Unified Architecture client connections to an OPC Unified Architecture server. This reduces the need for manual configuration for discovery and browsing mechanisms. This can be used for the configuration of communication networks based on the description of network configuration and structure (including communication components of sensors and actuators with respect to communication system parameters, network structure and wiring, quality of service, etc.).

Proposals for industrial usage and possible applications are listed in Annex A "Industrial application". Annex B "Mapping example" contains an informative mapping example to explain the applied mapping rules. Annex C contains the "AML Base Types" *NodeSet*¹⁾ including all *Nodes* and *References*, which are necessary for the application of AML and OPC Unified Architecture in combination.

This specification is based on the companion specification "OPC Unified Architecture for AutomationML" [13] which was created by a joint working group of the OPC Foundation and AutomationML e.V. It defines an OPC Unified Architecture information model to represent the AML models.

The tabular structure used within this specification is described in IEC 62541-5, 3.3, 4, and 5. The graphical notation used here is defined in IEC 62541-3, Annex D.

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.

ISO/IEC 25000:2014, *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE*

1) A *NodeSet* defines a set of *Nodes*, their *Attributes* and *References* (see IEC 62541-6, Annex F).

IEC 62541-1:2015, *OPC Unified Architecture — Part 1: Overview and concepts*

IEC 62541-2:2015, *OPC Unified Architecture — Part 2: Security Model*

IEC 62541-3:2015, *OPC Unified Architecture — Part 3: Address Space Model*

IEC 62541-5:2015, *OPC Unified Architecture — Part 5: Information Model*

IEC 62541-6:2015, *OPC Unified Architecture — Part 6: Mappings*

IEC 62541-7:2015, *OPC Unified Architecture — Part 7: Profiles*

IEC 62714-1:2014, *Engineering data exchange format for use in industrial automation systems engineering — Automation markup language — Part 1: Architecture and general requirements*

IEC 62714-2:2015, *Engineering data exchange format for use in industrial automation systems engineering — Automation markup language — Part 2: Role class libraries*

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

3 Terms and definitions

For the purposes of this document, the terms and definitions given in the IEC 62714 series and the IEC 62541 parts 1, 2, 3, 5, 6 and 7 apply.

AML-related terms and names are preceded by the prefix AutomationML.

4 Symbols and abbreviated terms

AML	Automation Markup Language
CAEX	Computer Aided Engineering eXchange
CIP	Continuous Improvement Processes
COLLADA	Collaborative Design Activity
CPPS	Cyber-Physical Production System
HMI	Human-Machine Interface
JT	Jupiter Tessellation
MCO	Manufacturing Change Order
MCR	Manufacturing Change Request
MRO	Maintenance, Repair, Overhaul
OLP	Offline programming
PLC	Programmable Logic Controller

STEP	Standard for the Exchange of Product model data
UA	Unified Architecture
XML	Extensible Markup Language

5 AML model mapping to OPC Unified Architecture information model

5.1 AML modelling concept

AML models are divided into different hierarchical trees (see Figure 1): InstanceHierarchy, RoleClassLib, SystemUnitClassLib, and InterfaceClassLib.

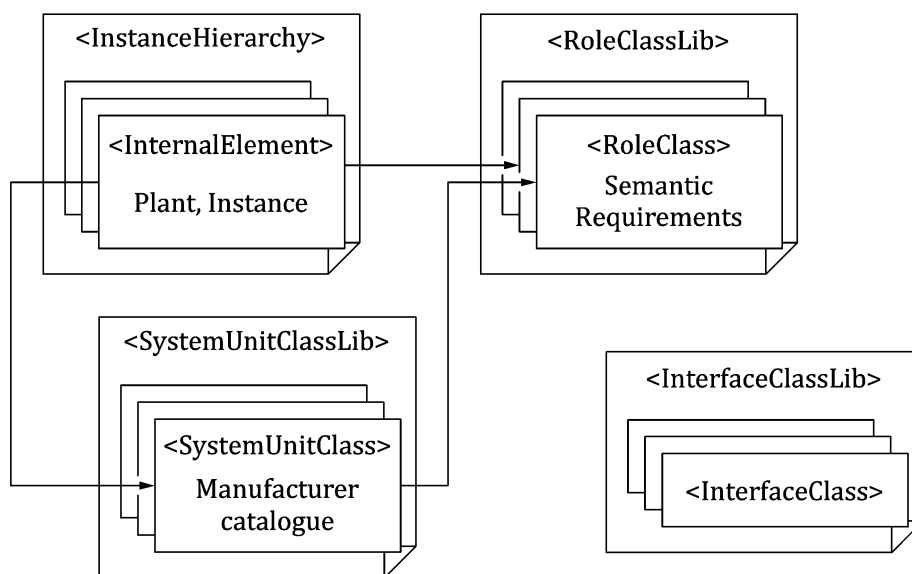


Figure 1 — AML hierarchical trees

The AML main elements and all possible relations between them are shown in Figure 2. The organization in a hierarchical tree is such that every element type can have child elements of the same type (see ChildElement relations). SystemUnitClasses, RoleClasses, and InterfaceClasses are defined in an inheritance structure; the RefBaseClassPath relations define such inheritance relations. RoleClasses can be assigned to SystemUnitClasses and InternalElements. In the latter case there are two different ways to make the RoleClass assignment using SupportedRoleClasses or the RoleRequirement. The remaining arrows in Figure 2 describe the possible relations between InternalElements and SystemUnitClasses. An InternalElement can be an instance of a SystemUnitClass. Within the scope of AML, SystemUnitClasses are only templates and can be changed after instantiation. The backward relation indicates that a SystemUnitClass can define sub-elements in terms of InternalElements. These sub-elements (InternalElements) should be included on instantiation. RoleClasses describe functions of a physical or logical plant object independent of a technical implementation. They offer a possibility of specifying an object in an abstract way and independent of the manufacturer. The assignment of a RoleClass to an element (SystemUnitClass or InternalElement) results in the allocation of fundamental functions or requirements.

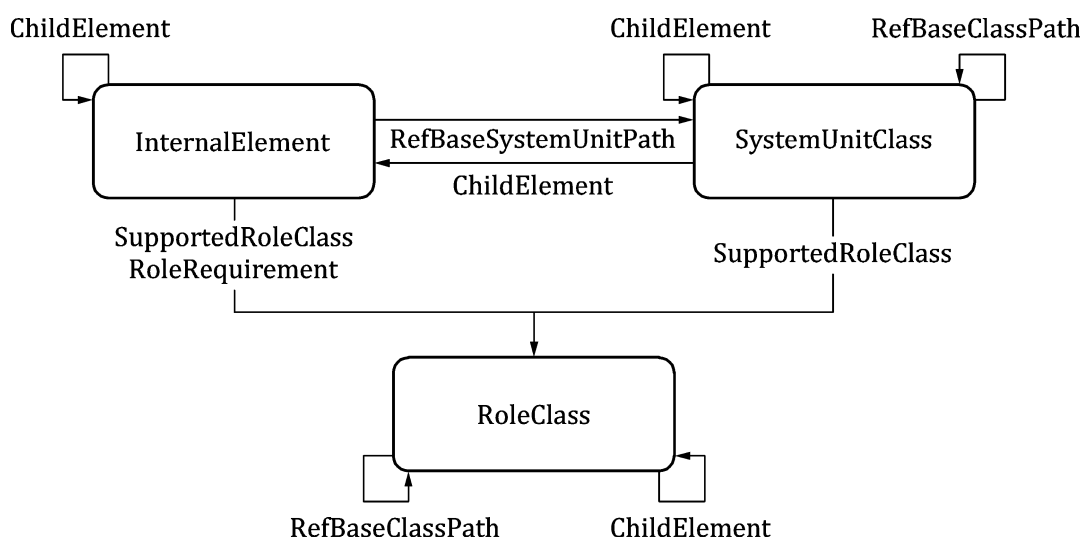


Figure 2 — AML main elements

Figure 3 depicts the basic structure of a RoleClass (RC), SystemUnitClass (SUC), or InternalElement (IE). Each can contain arbitrarily nested Attributes² and interfaces (see ChildElement relations). The interfaces are called ExternalInterfaces and are an instance of an InterfaceClass. The InterfaceClasses are stored hierarchically within InterfaceClassLibraries. They can be connected within an InstanceHierarchy to other ExternalInterfaces via InternalLinks. InterfaceClasses and certain ExternalInterfaces may also have arbitrarily nested Attributes.

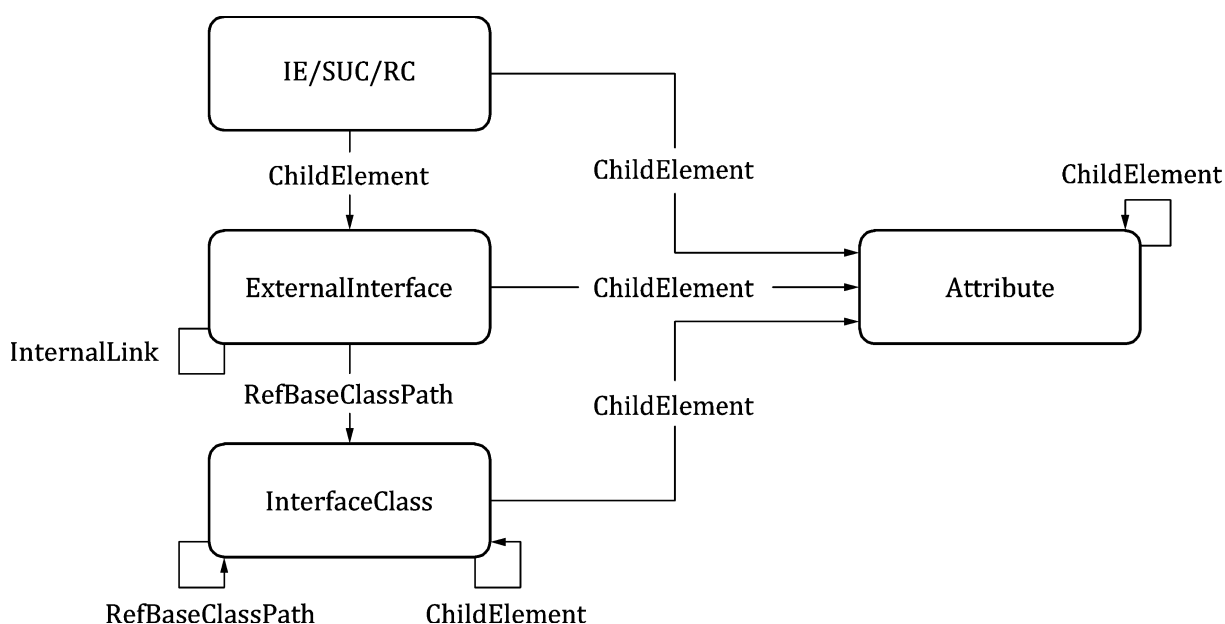


Figure 3 — AML main object details

² Attributes specify properties of an object, e.g. "length", according to IEC 62424.

5.2 How to map OPC Unified Architecture Namespaces

Several OPC Unified Architecture *NodeSets* (see Figure 4) are defined to organize the AddressSpace. Figure 4 includes the *NodeSet* name, the corresponding *Namespace URI*, and the file name containing the *NodeSet*. All *NodeSets* are based on the OPC Unified Architecture Base types. The *NodeSet* "AML base types" (see 5.5) defines all *Nodes* and type definitions necessary for the modelling of AML models in OPC Unified Architecture. This includes organizational *Nodes* for entry into the AML folder for *RoleClassLibs* or AML types (*ObjectType* "CAEXFileType"). One *NodeSet* is defined for the standardized, informative AML *libraries* as in IEC 62714-1, IEC 62714-2, IEC 62714-3, and IEC 62714-4. And finally, one or more *NodeSets* are used for the concrete AML documents.

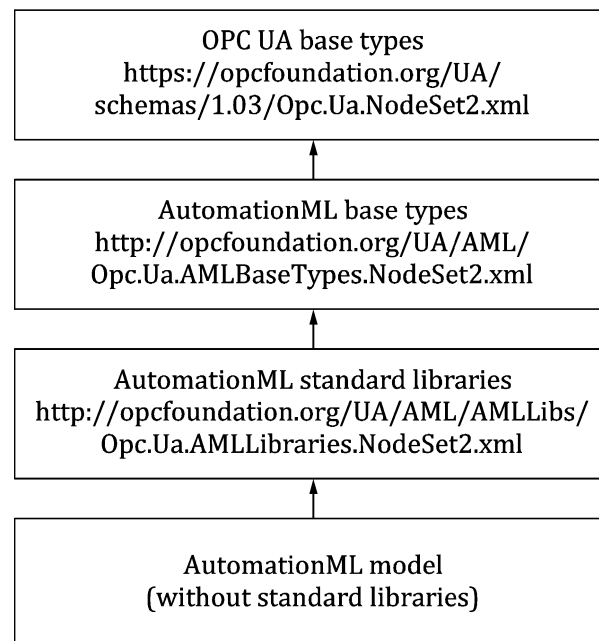


Figure 4 — Different *NodeSets* for the usage of AML in OPC Unified Architecture information models

Namespaces are used in OPC Unified Architecture to create unique identifiers across different naming authorities. The *Attributes NodeId* and *BrowseName* are identifiers. A *Node* in the OPC Unified Architecture *AddressSpace* is unambiguously identified using a *NodeId*. Unlike *NodeIds*, the *BrowseName* cannot be used to unambiguously identify a *Node*. Different *Nodes* may have the same *BrowseName*. They are used to build a browse path between two *Nodes* or to define a standard *Property*.

Servers may often choose to use the same *Namespace* for the *NodeId* and the *BrowseName*. However, if they want to provide a standard *Property*, its *BrowseName* will have the *Namespace* of the standards body, while the *Namespace* of the *NodeId* reflects something else, for example the EngineeringUnits *Property*. All *NodeIds* of *Nodes* not defined in this specification shall not use the standard *Namespaces*.

The *NodeIds* shall be generated during the mapping to OPC Unified Architecture following a specific schema which is arbitrary, e.g. numerical *NodeIds* from 1 to *n* (where *n* is the number of elements with *NodeId*).

NOTE For applications in embedded environments, numerical *NodeIds* or GUIDs should be used.

Table 1 provides a list of mandatory and optional *Namespaces* used in an AML OPC Unified Architecture *Server*.

Table 1 — Namespaces used in an OPC Unified Architecture server based on AML

Namespace	Description	Use
http://opcfoundation.org/UA/	<i>Namespace for <u>NodeIds</u> and <u>BrowseNames</u> defined in the OPC Unified Architecture specification. This <i>Namespace</i> shall have <i>Namespace</i> index 0.</i>	Mandatory
Local Server URI	<i>Namespace for <u>Nodes</u> defined in the local server. This may include types and instances used in a device represented by the server. This <i>Namespace</i> shall have <i>Namespace</i> index 1.</i>	Mandatory
http://opcfoundation.org/UA/AML/	<i>Namespace for <u>NodeIds</u> and <u>BrowseNames</u> defined in AML. The <i>Namespace</i> index shall be server-specific.</i>	Mandatory
http://opcfoundation.org/UA/AML/AMLLibs/	<i>Namespace for <u>NodeIds</u> and <u>BrowseNames</u> defined in AML <u>libraries</u>. The <i>Namespace</i> index is server-specific.</i>	Optional

5.3 How to map AML model elements

The main structure of an AML file as in Figure 1 is mapped into a slightly different structure as that depicted in Figure 5. The InstanceHierarchies and libraries shall be stored in common folders (“AutomationMLInstanceHierarchies”, “AutomationMLLibraries”, “AutomationMLSystemUnitClassLibs”, “AutomationMLRoleClassLibs”, “AutomationMLInterfaceClassLibs”).

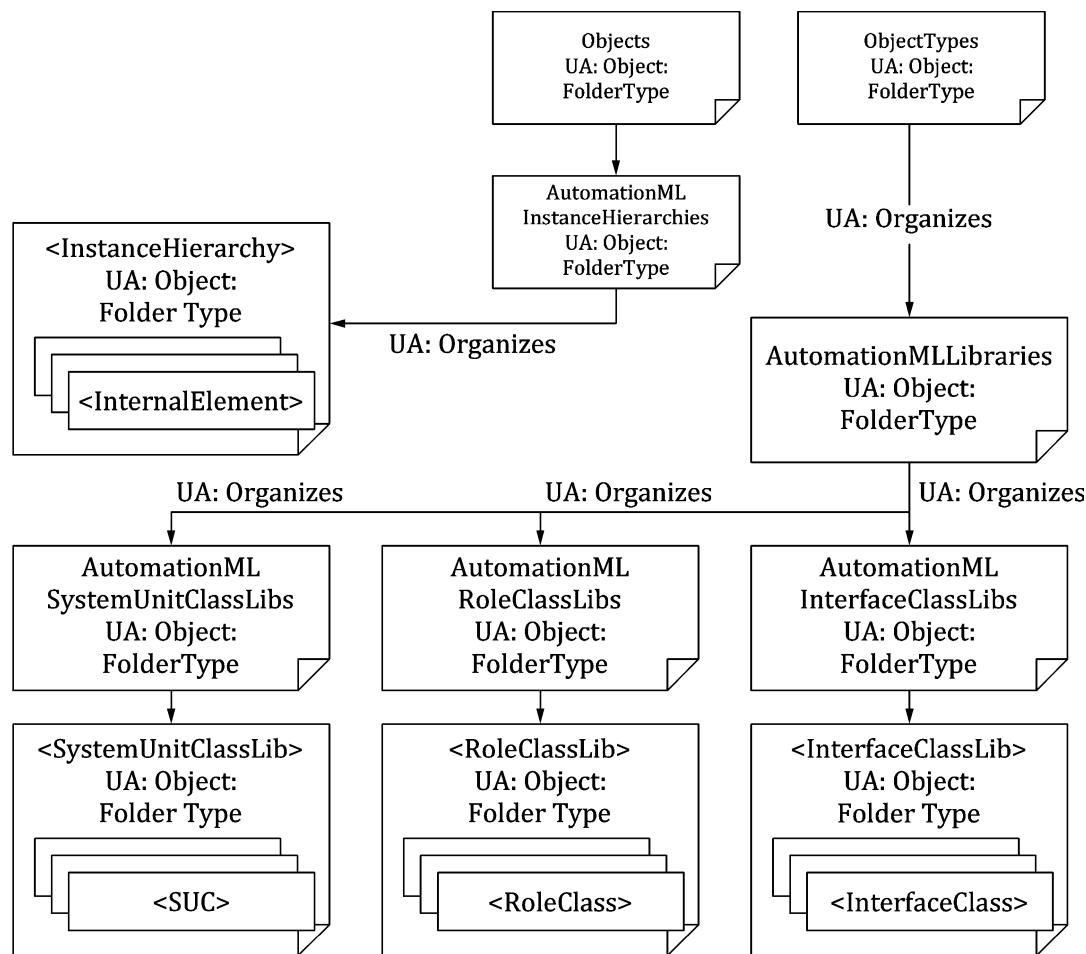


Figure 5 — CAEX file <-> OPC Unified Architecture Types [18]

The mapping of AML element type, and of their relations with OPC Unified Architecture *Nodes*, *NodeClasses* and *References* is depicted in Figure 6. The mapping of the internal structure of AML elements to OPC Unified Architecture *Nodes*, *NodeClasses* and *References* is depicted in Figure 7. The structure of the figures as compared to that in the figures in 5.1 has not been changed, so the mapping can be read directly from the graphic. Table 2, Table 3, and Table 4 summarize the mapping rules in lists. Basically, AML InternalElements are represented by OPC Unified Architecture *Objects*, while AML SystemUnitClasses and RoleClasses are represented by OPC Unified Architecture *ObjectTypes*. Detailed definitions of new *ObjectTypes*, *ReferenceTypes* and *VariableTypes* are given in 5.5.

The following provision applies to the mapping of AML models to OPC Unified Architecture models:

- For every *Reference* created in OPC Unified Architecture there shall be a corresponding *SourceNode* and *TargetNode*. Otherwise, *References* shall not be created.

ChildElement relations within class structures (SystemUnitClass, RoleClass, and InterfaceClass) are mapped to the OPC Unified Architecture *Organizes References*. InternalElements, which follow the ChildElement relation, are referenced via *HasComponent References*. The inheritance relations within SystemUnitClasses, RoleClasses, and InterfaceClasses are mapped to OPC Unified Architecture *HasSubtype References*. The internal structures of SystemUnitClass are depicted by InternalElements, which are connected via OPC Unified Architecture *HasComponent References*. The RefBaseClassPath relation in AML is an OPC Unified Architecture *HasSubtype Reference*. The RefBaseSystemUnitPath relation in AML is an OPC Unified

Architecture *HasTypeDefinition Reference*. The semantic direction of *HasSubtype Reference* is inverted compared to the corresponding *RefBaseClassPath* relation.

The referencing of *RoleClasses* from *InternalElements* or *SystemUnitClass* via *SupportedRoleClass* or *RoleRequirement* in AML is done via OPC Unified Architecture “HasAMLSupportedRoleClass” and “HasAMLRoleRequirement” *References*.

Table 2 — Element mapping

AML	OPC Unified Architecture
InternalElement	Object
SystemUnitClass	ObjectType
RoleClass	ObjectType

Table 3 — Attribute mapping

AML	OPC Unified Architecture
<u>Name</u>	<i>BrowseName</i>
<u>ID</u>	<i>Property “ID” with DataType String</i>
<u>Description</u>	<i>Description</i>
<u>Version</u>	<i>Property “Version”^a with DataType String</i>
<u>Revision</u>	— ^a
<u>Copyright</u>	<i>Property “Copyright” with DataType String</i>
<u>AdditionalInformation</u>	<i>Property “AdditionalInformation” with DataType String (ValueRank = 1)</i> Complete XML text, e.g. <AdditionalInformation AutomationMLVersion="2.0" />
<u>ChangeMode</u>	— ^a
<u>CAEXFile.FileName</u>	<i>Property “FileName” with DataType String^b</i>
<u>CAEXFile.SchemaVersion</u>	<i>Property “SchemaVersion” with DataType String</i>
<u>ExternalReference</u>	<i>Property Type “ExternalReferenceType” with DataType String</i>
<u>ExternalReference.Alias</u>	<i>BrowseName of the “ExternalReference” Property</i>
<u>ExternalReference.Path</u>	<i>Value of the “ExternalReference” Property</i>

<u>AML</u>	<u>OPC Unified Architecture</u>
<u>Attribute.Value</u>	<i>Value</i>
<u>Attribute.AttributeDataType</u>	<i>DataType</i> (see Table 7)
<u>Attribute.Unit</u>	<i>Property</i> "Unit" with <i>DataType String</i>
<u>Attribute.DefaultValue</u>	<i>Property</i> "DefaultValue" with same <i>DataType</i> as the <u>Attribute</u>
<u>Attribute.RefSemantic</u>	<i>Property</i> "RefSemantic" with <i>DataType String</i> (<i>ValueRank</i> = 1)
<u>Attribute.Constraint</u>	<i>Property</i> "Name of Constraint" with <i>DataType BaseDataType</i> and <i>VariableType</i> "OrdinalScaledConstraintType", "NominalScaledConstraintType" or "UnknownConstraintType"
<u>OrdinalScaledType.RequiredMaxValue</u>	"RequiredMaxValue" of <i>Property</i> of type "OrdinalConstraintType" with <i>DataType String</i>
<u>OrdinalScaledType.RequiredValue</u>	"RequiredValue" of <i>Property</i> of type "OrdinalConstraintType" with <i>DataType String</i>
<u>OrdinalScaledType.RequiredMinValue</u>	"RequiredMinValue" of <i>Property</i> of type "OrdinalConstraintType" with <i>DataType String</i>
<u>NominalScaledType.RequiredValue</u>	One item of <i>Property</i> of type "NominalConstraintType" with <i>DataType String</i> (<i>ValueRank</i> = 1)
<u>UnknownType.Requirements</u>	<i>Property</i> of type "UnknownConstraintType" with <i>DataType String</i>
<u>MappingObject</u>	<i>Reference</i> of type "HasAMLSupportedRoleClass", "HasAMLRoleRequirement" between the OPC UA <i>Nodes</i> of the mapped AML elements (<u>Attribute</u> or <u>ExternalInterface</u>).
CAEXBasicObject information (<u>Version</u> , <u>Description</u> , <u>Copyright</u> , and <u>AdditionalInformation</u>) of: <u>RefSemantic</u> , <u>RoleRequirement</u> , <u>SupportedRoleClass</u> , and <u>MappingObject</u>	— ^a
^a Currently not mapped into the OPC UA model. ^b For a better usability of the <u>CAEXFile.FileName</u> it should be used as part of the <i>BrowseName</i> .	

Table 4 — Relation mapping

<u>AML Source</u>	<u>AML Target</u>	<u>AML Relation</u>	<u>OPC Unified Architecture Reference</u>	<u>Description</u>
InternalElement	RoleClass	SupportedRoleClass	HasAMLSupport edRoleClass	Semantic similar to HasTypeDefinition
InternalElement	RoleClass	RoleRequirement	HasAMLRoleReq uirement	Semantic similar to HasTypeDefinition
InternalElement	SystemUnitClass	RefBaseSystemUnitPath	HasTypeDefiniti on	
InternalElement	InternalElement		HasComponent	ChildElement
SystemUnitClass	RoleClass	SupportedRoleClass	HasAMLSupport edRoleClass	Semantic similar to HasTypeDefinition
SystemUnitClass	InternalElement		HasComponent	ChildElement
SystemUnitClass	SystemUnitClass		Organizes	ChildElement
SystemUnitClass	SystemUnitClass	RefBaseClassPath	HasSubtype	
RoleClass	RoleClass		Organizes	ChildElement
RoleClass	RoleClass	RefBaseClassPath	HasSubtype	

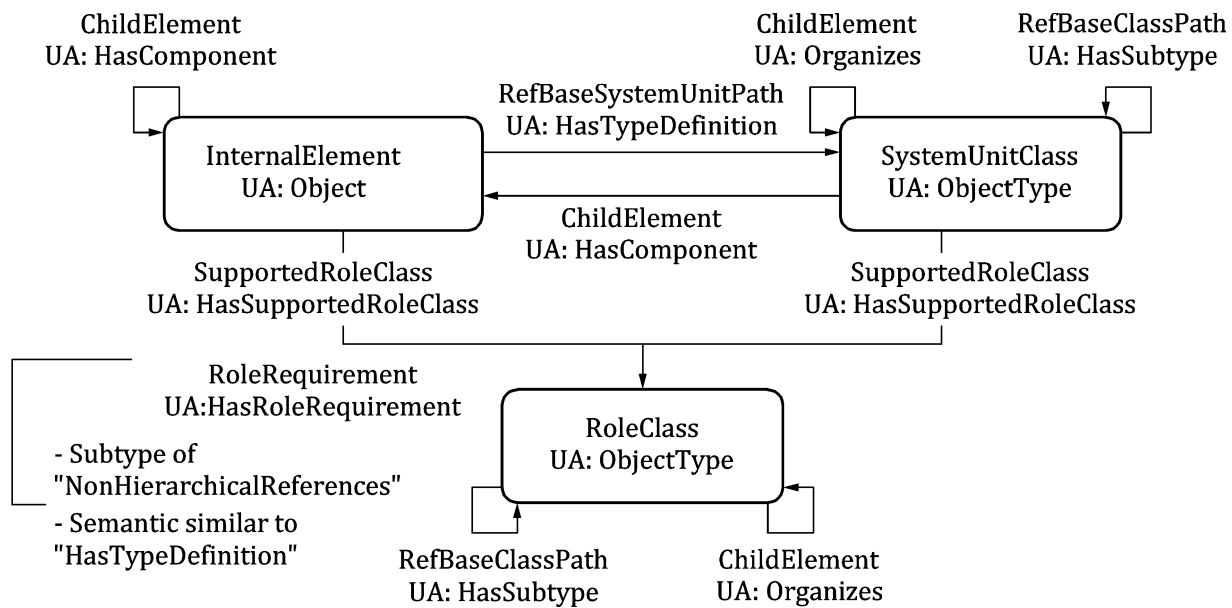


Figure 6 — AML element types and the OPC Unified Architecture *Nodes and References* [18]

The mappings of the AML object details (see Figure 3) are depicted in Figure 7. The analogy between the two figures is apparent. Table 5 and Table 6 summarize the mapping rules in lists. AML Attributes become OPC Unified Architecture *Variables*. AML InterfaceClasses are transformed into OPC Unified Architecture *ObjectTypes* and the corresponding AML ExternalInterface elements are modelled by OPC Unified Architecture Objects.

The Attributes and ExternalInterfaces are referenced via *HasComponent References*. The InterfaceClass is the type definition of the ExternalInterface; therefore, it is assigned via *HasTypeDefinition Reference*. Relations between ExternalInterfaces (AML InternalLinks) are assigned via “HasAMLInternalLink” *Reference*. Details are given in 5.4.

Table 5 — Element mapping

AML	OPC Unified Architecture
InternalElement	Object
SystemUnitClass	ObjectType
RoleClass	ObjectType
InterfaceClass	ObjectType
Attribute	Variable
ExternalInterface	Object

Table 6 — Relation mapping

<u>AML Source</u>	<u>AML Target</u>	<u>AML Relation</u>	<u>OPC Unified Architecture Reference</u>	<u>Description</u>
InternalElement	Attribute		HasComponent	ChildElement
SystemUnitClass	Attribute		HasComponent	ChildElement
RoleClass	Attribute		HasComponent	ChildElement
InternalElement	ExternalInterface		HasComponent	ChildElement
SystemUnitClass	ExternalInterface		HasComponent	ChildElement
RoleClass	ExternalInterface		HasComponent	ChildElement
ExternalInterface	ExternalInterface	InternalLink	HasAMLInternalLink	
ExternalInterface	Attribute		HasComponent	ChildElement
ExternalInterface	InterfaceClass	RefBaseClassPath	HasTypeDefinition	
InterfaceClass	Attribute		HasComponent	ChildElement
InterfaceClass	InterfaceClass	RefBaseClassPath	HasSubtype	
InterfaceClass	InterfaceClass		Organizes	ChildElement
Attribute	Attribute		HasComponent	ChildElement

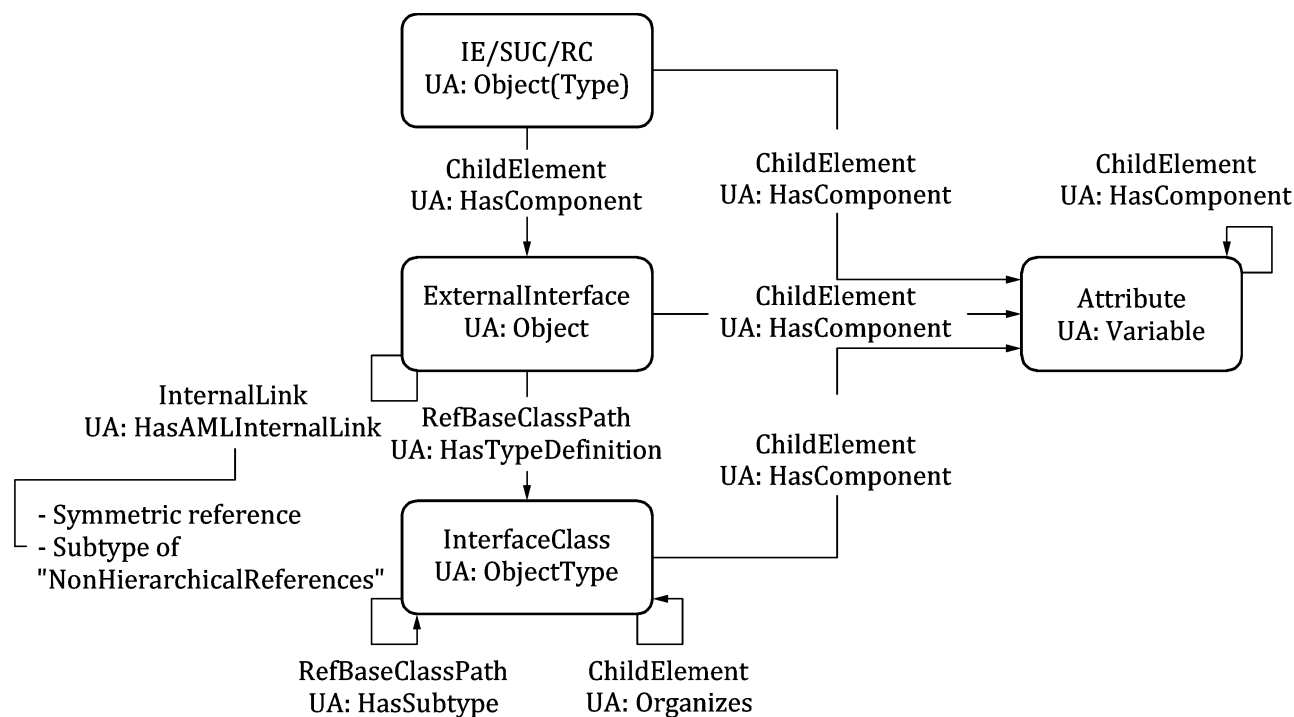


Figure 7 — Internal structure of AML elements and the OPC Unified Architecture Nodes and References [18]

5.4 Mapping of AML XML DataTypes to OPC Unified Architecture DataTypes

The mapping of AML XML DataTypes to OPC Unified Architecture *DataTypes* is described in Table 7.

Table 7 — Mapping of XML DataTypes to OPC Unified Architecture DataTypes

<u>XML SimpleType</u>	<u>OPC Unified Architecture DataType</u>
string	String
boolean	Boolean
decimal	Double
float	Float
double	Double
duration	Duration
dateTime	DateTime
time	Duration
date	DateTime
gYearMonth	DateString

<u>XML SimpleType</u>	<u>OPC Unified Architecture DataType</u>
gYear	DateString
gMonthDay	String
gDay	String
gMonth	String
hexBinary	ByteString
base64Binary	ByteString
anyURI	String
QName	String
NOTATION	String (ValueRank = 1)
normalizedString	String
token	String
language	LocaleId
NMTOKEN	String
NMTOKENS	String (ValueRank = 1)
Name	String
NCName	String
ID	String
IDREF	String
IDREFS	String (ValueRank = 1)
ENTITY	String
ENTITIES	String (ValueRank = 1)
integer	Int64
nonPositiveInteger	Int64
negativeInteger	Int64
long	Int64

<u>XML SimpleType</u>	<u>OPC Unified Architecture DataType</u>
int	Int32
short	Int16
byte	SByte
nonNegativeInteger	UInt64
unsignedLong	UInt64
unsignedInt	UInt32
unsignedShort	UInt16
unsignedByte	Byte
positiveInteger	Int64

5.5 AML base types for OPC Unified Architecture information model

5.5.1 ObjectTypes

5.5.1.1 CAEXBasicObjectType

5.5.1.1.1 General

The “CAEXBasicObjectType” defines all general characteristics of a CAEX element. All other CAEX elements derive from the “CAEXBasicObjectType”. The “CAEXBasicObjectType” inherits all *Properties* of the *BaseObjectType*.

5.5.1.1.2 ObjectType Definition

The “CAEXBasicObjectType” is formally defined in Table 8.

Table 8 — CAEXBasicObjectType Definition

Attribute	Value				
BrowseName	CAEXBasicObjectType				
IsAbstract	True				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the BaseObjectType					
HasProperty	Variable	Version	String	PropertyType	Optional
HasProperty	Variable	Copyright	String	PropertyType	Optional
HasProperty	Variable	AdditionalInformation	String (ValueRank = 1)	PropertyType	Optional

5.5.1.1.3 ObjectType Description

5.5.1.1.3.1 Version

“Version” provides the optional organizational information about the state of the version for the “CAEXBasicObjectType”

5.5.1.1.3.2 Copyright

“Copyright” provides the optional organizational information about copyright for the “CAEXBasicObjectType”

5.5.1.1.3.3 AdditionalInformation

“AdditionalInformation” provides an optional auxiliary field that may contain any additional information about a CAEX object defined by the “CAEXBasicObjectType”. The *ValueRank* of this *Property* is 1.

5.5.1.2 CAEXFileType

5.5.1.2.1 General

The “CAEXFileType” defines all general characteristics of a CAEX file and includes all CAEX libraries and instance hierarchies. The “CAEXFileType” inherits all *Properties* of the “CAEXBasicObjectType”.

5.5.1.2.2 ObjectType Definition

The “CAEXFileType” is formally defined in Table 9.