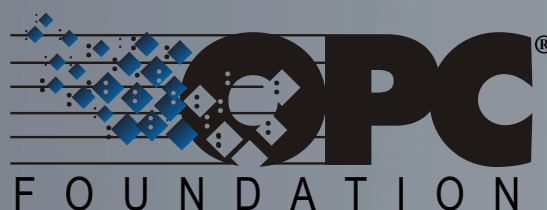




<AutomationML/>

**The Glue for Seamless
Automation Engineering**



**AutomationML Whitepaper:
OPC Unified Architecture
Information Model for
AutomationML**

State: March 2016

© AutomationML consortium, OPC Foundation
Version March 2016

Contact: www.automationml.org

Table of content

Table of content	3
List of Figures	8
List of Tables	9
AGREEMENT OF USE	13
COPYRIGHT RESTRICTIONS	13
PATENTS	13
WARRANTY AND LIABILITY DISCLAIMERS	13
RESTRICTED RIGHTS LEGEND	14
COMPLIANCE	14
TRADEMARKS	14
GENERAL PROVISIONS	14
1 Introduction.....	15
1.1 Scope	15
1.2 Reference documents	15
2 Terms, definitions, and conventions	16
2.1 Use of terms	16
2.2 Abbreviations and symbols	16
2.3 Conventions used in this document.....	16
2.3.1 Conventions for naming of interconnection between elements.....	16
2.3.2 Conventions for Node descriptions.....	17
2.3.3 NodeIds and BrowseNames	19
2.3.3.1 NodeIds.....	19
2.3.3.2 BrowseNames	19
2.3.4 Common Attributes	20
2.3.4.1 General	20
2.3.4.2 Objects.....	20
2.3.4.3 Variables.....	20
2.3.4.4 VariableTypes	21
3 General information to AutomationML and OPC UA	22
3.1 Introduction to AutomationML	22
3.1.1 Top-level format CAEX.....	23
3.2 Introduction to OPC Unified Architecture	24
3.2.1 General.....	24
3.2.2 Graphical Notation	26
3.3 Use Cases	27
4 AutomationML Model Overview	28
4.1 Modeling concepts	28
4.2 Model Overview	30
4.2.1 Mapping Explanations for Namespaces.....	30
4.2.2 Mapping Explanations for AutomationML model elements	30
4.3 Mapping Example	35
5 AutomationML Base Types OPC UA Model	38
5.1 ObjectTypes	38
5.1.1 General.....	38
5.1.2 CAEXBasicObjectType.....	38

5.1.2.1	General	38
5.1.2.2	ObjectType Definition	38
5.1.2.3	ObjectType Description	39
5.1.3	CAEXFileType	39
5.1.3.1	General	39
5.1.3.2	ObjectType Definition	39
5.1.3.3	ObjectType Description	39
5.1.4	CAEXObjectType	40
5.1.4.1	General	40
5.1.4.2	ObjectType Definition	40
5.1.4.3	ObjectType Description	40
5.1.5	AutomationMLBaseInterface	40
5.1.5.1	General	40
5.1.5.2	ObjectType Definition	40
5.1.6	AutomationMLBaseRole	40
5.1.6.1	General	40
5.1.6.2	ObjectType Definition	40
5.1.7	AutomationMLBaseSystemUnit	41
5.1.7.1	General	41
5.1.7.2	ObjectType Definition	41
5.2	ReferenceTypes	41
5.2.1	HasAMLRoleReference	41
5.2.1.1	General	41
5.2.1.2	ObjectType Definition	41
5.2.2	HasAMLInternalLink	42
5.2.2.1	General	42
5.2.2.2	ReferenceType Definition	42
5.3	VariableTypes.....	42
5.3.1	AMLBaseVariableType	42
5.3.1.1	General	42
5.3.1.2	VariableType Definition	42
5.3.1.3	VariableType Description.....	42
5.4	Mapping of AutomationML XML DataTypes to OPC UA DataTypes	43
5.4.1	OPC UA Objects used to organize the address space structure	43
5.4.1.1	Instances and entry points.....	45
6	AutomationML Libraries OPC UA Model.....	46
6.1	General	46
6.2	AutomationMLInterfaceClassLib.....	46
6.2.1	Instances	46
6.2.1.1	AutomationMLInterfaceClassLib	46
6.2.2	ObjectTypes.....	46
6.2.2.1	AutomationMLBaseInterface	46
6.2.2.2	AttachmentInterface	47
6.2.2.3	Communication.....	47
6.2.2.4	SignalInterface	47
6.2.2.5	ExternalDataConnector	48
6.2.2.6	COLLADAInterface	48
6.2.2.7	PLCopenXMLInterface.....	49
6.2.2.8	LogicInterface.....	49

6.2.2.9	VariableInterface	49
6.2.2.10	InterlockingVariableInterface	50
6.2.2.11	InterlockingConnector	50
6.2.2.12	Order	51
6.2.2.13	PortConnector	51
6.2.2.14	PPRConnector	52
6.3	AutomationMLBaseRoleClassLib	52
6.3.1	Instances	52
6.3.1.1	AutomationMLBaseRoleClassLib	52
6.3.2	ObjectTypes	52
6.3.2.1	AutomationMLBaseRoleClassLib	52
6.3.2.2	Group	53
6.3.2.3	Facet	53
6.3.2.4	Port	54
6.3.2.5	Resource	55
6.3.2.6	Product	55
6.3.2.7	Process	55
6.3.2.8	Structure	56
6.3.2.9	ResourceStructure	56
6.3.2.10	ProductStructure	56
6.3.2.11	ProcessStructure	57
6.3.2.12	Frame	57
6.3.2.13	ComponentGroup	57
6.3.2.14	SignalGroup	58
6.3.2.15	PropertySet	58
6.4	AutomationMLDMIRoleClassLib	59
6.4.1	Instances	59
6.4.1.1	AutomationMLDMIRoleClassLib	59
6.4.2	ObjectTypes	59
6.4.2.1	DiscManufacturingEquipment	59
6.4.2.2	Transport	59
6.4.2.3	Storage	60
6.4.2.4	Fixture	60
6.4.2.5	Gate	60
6.4.2.6	Robot	61
6.4.2.7	Tool	61
6.4.2.8	Carrier	61
6.4.2.9	Machine	62
6.4.2.10	StaticObject	62
6.5	AutomationMLCMIRoleClassLib	62
6.5.1	Instances	62
6.5.2	ObjectTypes	63
6.5.2.1	ContManufacturingEquipment	63
6.6	AutomationMLBMIRoleClassLib	63
6.6.1	Instances	63
6.6.1.1	AutomationMLBMIRoleClassLib	63
6.6.2	ObjectTypes	63
6.6.2.1	BatchManufacturingEquipment	63
6.7	AutomationMLCSRoleClassLib	64

6.7.1	Instances	64
6.7.1.1	AutomationMLCSRoleClassLib	64
6.7.2	ObjectTypes	64
6.7.2.1	ControlEquipment	64
6.7.2.2	Communication	64
6.7.2.3	ControlHardware	65
6.7.2.4	PC	65
6.7.2.5	IPC	65
6.7.2.6	Handheld	66
6.7.2.7	EmbeddedDevice	66
6.7.2.8	Sensor	66
6.7.2.9	Actuator	67
6.7.2.10	Controller	67
6.7.2.11	PLC	67
6.7.2.12	NC	68
6.7.2.13	RC	68
6.7.2.14	PAC	69
6.8	AutomationMLExtendedRoleClassLib	69
6.8.1	Instances	69
6.8.1.1	AutomationMLExtendedRoleClassLib	69
6.8.2	ObjectTypes	69
6.8.2.1	PLCFacet	69
6.8.2.2	HMIFacet	70
6.8.2.3	Enterprise	70
6.8.2.4	Site	71
6.8.2.5	Area	71
6.8.2.6	ProductionLine	72
6.8.2.7	WorkCell	72
6.8.2.8	ProcessCell	73
6.8.2.9	Unit	73
6.8.2.10	ProductionUnit	74
6.8.2.11	StorageZone	74
6.8.2.12	StorageUnit	75
6.8.2.13	Turntable	75
6.8.2.14	Conveyor	76
6.8.2.15	BeltConveyor	76
6.8.2.16	RollConveyor	76
6.8.2.17	ChainConveyor	77
6.8.2.18	PalletConveyor	77
6.8.2.19	OverheadConveyor	77
6.8.2.20	LiftingTable	78
6.8.2.21	AGV	78
6.8.2.22	Transposer	79
6.8.2.23	CarrierHandlingSystem	79
6.8.2.24	BodyStore	79
6.8.2.25	Lift	80
6.8.2.26	Rollerbed	80
6.8.2.27	StationaryTool	80
6.8.2.28	MovableTool	81

6.8.2.29	ControlCabinet	81
6.8.2.30	IODevice	81
6.8.2.31	HMI	82
6.8.2.32	WarningEquipment	82
6.8.2.33	ActuatingDrive	82
6.8.2.34	MotionController	83
6.8.2.35	Panel	83
6.8.2.36	MeasuringEquipment	84
6.8.2.37	Clamp	84
6.8.2.38	ProcessController	84
6.8.2.39	Loader	85
6.8.2.40	Unloader	85
7	Profiles	86
7.1	OPC UA Conformance Units and Profiles	86
8	Address space structure	86
8.1	Handling of OPC UA namespaces	86
Annex A	(informative): Mapping example	88
Annex B	(informative): Bibliography	95

List of Figures

<i>Figure 1: AutomationML base structure</i>	<i>23</i>
<i>Figure 2: AutomationML topology description architecture</i>	<i>24</i>
<i>Figure 3: OPC UA Graphical Notation for NodeClasses</i>	<i>26</i>
<i>Figure 4: OPC UA Graphical Notation for References</i>	<i>26</i>
<i>Figure 5: OPC UA Graphical Notation Example</i>	<i>27</i>
<i>Figure 6: Goals of AutomationML and OPC UA</i>	<i>27</i>
<i>Figure 7: AutomationML hierarchical trees</i>	<i>28</i>
<i>Figure 8: AutomationML main elements</i>	<i>29</i>
<i>Figure 9: AutomationML main object details.</i>	<i>29</i>
<i>Figure 10: Different nodesets for the usage of AutomationML in OPC UA information models</i>	<i>30</i>
<i>Figure 11: CAEX file <-> OPC UA Types [1]</i>	<i>31</i>
<i>Figure 12: AutomationML BaseElementTypes and the OPC UA Types [1]</i>	<i>33</i>
<i>Figure 13: AutomationML Element and the OPC UA Types [1]</i>	<i>34</i>
<i>Figure 14: AutomationML example.</i>	<i>35</i>
<i>Figure 15: Example main structure in OPC UA.</i>	<i>36</i>
<i>Figure 16: Example role classes in OPC UA.</i>	<i>36</i>
<i>Figure 17: Example system unit classes in OPC UA.</i>	<i>37</i>
<i>Figure 18: Example interface classes in OPC UA.</i>	<i>37</i>
<i>Figure 19: Example InstanceHierarchy in OPC UA.</i>	<i>38</i>
<i>Figure 20: AutomationML information model structure in OPC UA</i>	<i>44</i>
<i>Figure 21: AutomationML XML text</i>	<i>89</i>
<i>Figure 22: OPC UA XML text</i>	<i>94</i>

List of Tables

Table 1: Type Definition Table.....	17
Table 2: Examples of DataTypes	18
Table 3: Common Node Attributes	20
Table 4: Common Object Attributes	20
Table 5: Common Variable Attributes	20
Table 6: Common VariableType Attributes.....	21
Table 7: Element mapping.....	32
Table 8: Relation mapping.....	32
Table 9: Element mapping.....	33
Table 10: Relation mapping.....	34
Table 11: CAEXBasicObjectType Definition.....	39
Table 12: CAEXFileType Definition	39
Table 13: CAEXObjectType Definition	40
Table 14: AutomationMLBaseInterface Definition	40
Table 15: AutomationMLBaseRole Definition	41
Table 16: AutomationMLBaseSystemUnit Definition	41
Table 17: HasAMLRoleReference.....	41
Table 18: HasAMLInternalLink.....	42
Table 19: AMLBaseVariableType Definition.....	42
Table 20: Mapping of XML data types to OPC UA BaseTypes	43
Table 21: AutomationMLFiles Instance Definition	45
Table 22: AutomationMLInstanceHierarchies Instance Definition	45
Table 23: AutomationMLLibraries Instance Definition	45
Table 24: InterfaceClassLibs Instance Definition	45
Table 25: RoleClassLibs Instance Definition.....	46
Table 26: SystemUnitClassLibs Instance Definition	46
Table 27: AutomationMLInterfaceClassLib Instance Definition	46
Table 28: AutomationMLBaseInterface Definition	47
Table 29: AttachmentInterface Definition	47
Table 30: Communication Definition	47
Table 31: SignalInterface Definition.....	48
Table 32: ExternalDataConnector Definition	48
Table 33: COLLADAInterface Definition.....	49
Table 34: PLCopenXMLInterface Definition	49
Table 35: LogicInterface Definition	49
Table 36: VariableInterface Definition.....	50
Table 37: InterlockingVariableInterface Definition.....	50
Table 38: InterlockingConnector Definition	51
Table 39: Order Definition	51
Table 40: PortConnector Definition.....	52
Table 41: PPRConnector Definition	52

Table 42: AutomationMLBaseRoleClassLib Instance Definition	52
Table 43: AutomationMLBaseRole Definition	53
Table 44: Group Definition	53
Table 45: Facet Definition	54
Table 46: Port Definition	54
Table 47: Resource Definition	55
Table 48: Product Definition	55
Table 49: Process Definition	56
Table 50: Structure Definition	56
Table 51: ResourceStructure Definition	56
Table 52: ProductStructure Definition	57
Table 53: ProcessStructure Definition	57
Table 54: Frame Definition	57
Table 55: ComponentGroup Definition	58
Table 56: SignalGroup Definition	58
Table 57: PropertySet Definition	59
Table 58: AutomationMLDMIRoleClassLib Instance Definition	59
Table 59: DiscManufacturingEquipment Definition	59
Table 60: Transport Definition	60
Table 61: Storage Definition	60
Table 62: Fixture Definition	60
Table 63: Gate Definition	61
Table 64: Robot Definition	61
Table 65: Tool Definition	61
Table 66: Carrier Definition	62
Table 67: Machine Definition	62
Table 68: StaticObject Definition	62
Table 69: AutomationMLCMIRoleClassLib Instance Definition	63
Table 70: ContManufacturingEquipment Definition	63
Table 71: AutomationMLBMIRoleClassLib Instance Definition	63
Table 72: BatchManufacturingEquipment Definition	64
Table 73: AutomationMLCSRoleClassLib Instance Definition	64
Table 74: ControlEquipment Definition	64
Table 75: Communication Definition	65
Table 76: ControlHardware Definition	65
Table 77: PC Definition	65
Table 78: IPC Definition	66
Table 79: Handheld Definition	66
Table 80: EmbeddedDevice Definition	66
Table 81: Sensor Definition	67
Table 82: Actuator Definition	67
Table 83: Controller Definition	67
Table 84: PLC Definition	68

Table 85: NC Definition	68
Table 86: RC Definition	69
Table 87: PAC Definition	69
Table 88: AutomationMLExtendedRoleClassLib Instance Definition	69
Table 89: PLCFacet Definition	70
Table 90: HMIFacet Definition	70
Table 91: Enterprise Definition	71
Table 92: Site Definition	71
Table 93: Area Definition	72
Table 94: ProductionLine Definition	72
Table 95: WorkCell Definition	73
Table 96: ProcessCell Definition	73
Table 97: Unit Definition	74
Table 98: ProductionUnit Definition	74
Table 99: StorageZone Definition	75
Table 100: StorageUnit Definition	75
Table 101: Turntable Definition	76
Table 102: Conveyor Definition	76
Table 103: BeltConveyor Definition	76
Table 104: RollConveyor Definition	77
Table 105: ChainConveyor Definition	77
Table 106: PalletConveyor Definition	77
Table 107: OverheadConveyor Definition	78
Table 108: LiftingTable Definition	78
Table 109: AGV Definition	79
Table 110: Transposer Definition	79
Table 111: CarrierHandlingSystem Definition	79
Table 112: BodyStore Definition	80
Table 113: Lift Definition	80
Table 114: Rollerbed Definition	80
Table 115: StationaryTool Definition	81
Table 116: MovableTool Definition	81
Table 117: ControlCabinet Definition	81
Table 118: IODevice Definition	82
Table 119: HMI Definition	82
Table 120: WarningEquipment Definition	82
Table 121: ActuatingDrive Definition	83
Table 122: MotionController Definition	83
Table 123: Panel Definition	84
Table 124: MeasuringEquipment Definition	84
Table 125: Clamp Definition	84
Table 126: ProcessController Definition	85
Table 127: Loader Definition	85

<i>Table 128: Unloader Definition</i>	<i>86</i>
<i>Table 129: AutomationML Server Facet Definition</i>	<i>86</i>
<i>Table 130: AutomationML Client Facet Definition</i>	<i>86</i>
<i>Table 131: Namespaces used in a AutomationML Server</i>	<i>87</i>

AGREEMENT OF USE

COPYRIGHT RESTRICTIONS

- This document is provided "as is" by the OPC Foundation and the AutomationML e.V..
- Right of use for the Mapping Document is restricted to the Mapping Document.
- Right of use for the Mapping Document will be granted without cost.
- The Mapping Document may be distributed through computer systems, printed or copied as long as the content remains unchanged and the document is not modified.
- OPC Foundation and AutomationML e.V. do not guarantee usability for any purpose and shall not be made liable for any case using the content of this document.
- The user of the document agrees to indemnify OPC Foundation and AutomationML e.V. and their officers, directors and agents harmless from all demands, claims, actions, losses, damages (including damages from personal injuries), costs and expenses (including attorneys' fees) which are in any way related to activities associated with its use of content from the Mapping Document.
- The document shall not be used in conjunction with company advertising, shall not be sold or licensed to any party.
- The intellectual property and copyright is solely owned by the OPC Foundation and AutomationML e.V..

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OPC or AutomationML specifications may require use of an invention covered by patent rights. OPC Foundation or AutomationML e.V. shall not be responsible for identifying patents for which a license may be required by any OPC or AutomationML specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OPC or AutomationML specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

WARRANTY AND LIABILITY DISCLAIMERS

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. THE OPC FOUNDATION NOR AUTOMATIONML E.V. MAKES NO WARRANTY OF ANY KIND, EXPRESSED OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE. IN NO EVENT SHALL THE OPC FOUNDATION NOR AUTOMATIONML E.V. BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The entire risk as to the quality and performance of software developed using this specification is borne by you.

RESTRICTED RIGHTS LEGEND

This Specification is provided with Restricted Rights. Use, duplication or disclosure by the U.S. government is subject to restrictions as set forth in (a) this Agreement pursuant to DFARs 227.7202-3(a); (b) subparagraph (c)(1)(i) of the Rights in Technical Data and Computer Software clause at DFARs 252.227-7013; or (c) the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 subdivision (c)(1) and (2), as applicable. Contractor / manufacturer are the OPC Foundation, 16101 N. 82nd Street, Suite 3B, Scottsdale, AZ, 85260-1830

COMPLIANCE

The combination of AutomationML e.V. and OPC Foundation shall at all times be the sole entities that may authorize developers, suppliers and sellers of hardware and software to use certification marks, trademarks or other special designations to indicate compliance with these materials as specified within this document. Products developed using this specification may claim compliance or conformance with this specification if and only if the software satisfactorily meets the certification requirements set by AutomationML e.V. or the OPC Foundation. Products that do not meet these requirements may claim only that the product was based on this specification and must not claim compliance or conformance with this specification.

TRADEMARKS

Most computer and software brand names have trademarks or registered trademarks. The individual trademarks have not been listed here.

GENERAL PROVISIONS

Should any provision of this Agreement be held to be void, invalid, unenforceable or illegal by a court, the validity and enforceability of the other provisions shall not be affected thereby.

This Agreement shall be governed by and construed under the laws of Germany.

This Agreement embodies the entire understanding between the parties with respect to, and supersedes any prior understanding or agreement (oral or written) relating to, this specification.

1 Introduction

1.1 Scope

This specification was created by a joint working group of the OPC Foundation and AutomationML e.V.. It defines an OPC UA Information Model to represent the AutomationML models.

OPC Foundation

The OPC Foundation defines standards for online data exchange between automation systems. They address access to current data (OPC DA), alarms and events (OPC A&E) and historical data (OPC HDA). Those standards are successfully applied in industrial automation.

The new OPC Unified Architecture (OPC UA) unifies the existing standards and brings them to state-of-the-art technology using service-oriented architecture (SOA). Platform-independent technology allows the deployment of OPC UA beyond current OPC applications only running on Windows-based PC systems. OPC UA can also run on embedded systems as well as Linux / UNIX based enterprise systems. The provided information can be generically modelled and therefore arbitrary information models can be provided using OPC UA.

AutomationML e.V.

The AutomationML initiative is an open, in 2006 founded industrial consortium which became a registered association in 2009. It welcomes all interested company and research institute.

Purposes of AutomationML e.V. are promotion and further development of AutomationML to standardise data exchange in the engineering process of production systems. Therefore, AutomationML e.V. develops and maintains an open, neutral, XML based, and free industry data representation standard which enables a domain and company spanning transfer of engineering data. Instead of developing a new data format for that purpose, already existing formats were used, extended, adapted, and merged in a proper way. So far, the representation of plant specific data in general and in special the plant structure, geometry and kinematics, and logic description is possible. Additional representations for networks, mechatronical systems, and others are in progress. Hence, AutomationML is the most comprehensive data format of plant engineering. It is already used in the field and is also available in several products. Within IEC 62714 all parts of AutomationML are going to be standardised internationally.

1.2 Reference documents

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

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

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

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

IEC 62541-100:2015, *OPC Unified Architecture: Part 100: Device interface*

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

IEC 62714-4, *Engineering data exchange format for use in industrial automation systems engineering: Automation markup language: Part 3: Logic and behavior*

IEC 62264-1: 2013, *Enterprise-control system integration: Part 1: Models and terminology (similar to ANSI ISA 95)*

IEC 61512-1:1997, *Batch control: Part 1: Models and terminology (similar to ANSI ISA 88)*

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

2 Terms, definitions, and conventions

2.1 Use of terms

Defined terms of OPC UA specifications, types and their components defined in OPC UA specifications and in this specification are highlighted with *italic* in this document.

AutomationML related terms and names are always used together with the prefix AutomationML.

2.2 Abbreviations and symbols

AML	Automation Markup Language
CAEX	Computer Aided Engineering Exchange
HMI	Human-Machine Interface
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
UA	Unified Architecture
XML	Extensible Markup Language

2.3 Conventions used in this document

2.3.1 Conventions for naming of interconnection between elements

Relations between elements in AutomationML are referred to as references between nodes in OPC UA.

2.3.2 Conventions for Node descriptions

Node definitions are specified using tables (See Table 1)

Attribute	Value				
Attribute name	Attribute value. If it is an optional Attribute that is not set "--" will be used.				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
ReferenceType name	<i>NodeClass</i> of the <i>TargetNode</i> .	<i>BrowseName</i> of the target <i>Node</i> . If the <i>Reference</i> is to be instantiated by the server, then the value of the target <i>Node</i> 's <i>BrowseName</i> is "--".	<i>Attributes</i> of the referenced <i>Node</i> , only applicable for <i>Variables</i> and <i>Objects</i> .		Referenced <i>ModellingRule</i> of the referenced <i>Object</i> .
Notes – Notes referencing footnotes of the table content.					

Table 1: Type Definition Table

Attributes are defined by providing the *Attribute* name and a value, or a description of the value.

References are defined by providing the *ReferenceType* name, the *BrowseName* of the *TargetNode* and its *NodeClass*.

- If the *TargetNode* is a component of the *Node* being defined in the table the *Attributes* of the composed *Node* are defined in the same row of the table. That implies that the referenced *Node* has a *HasModelParent Reference* with the *Node* defined in the Table as *TargetNode* (see IEC 62541-3:2015) for the definition of *ModelParents*).
- The *DataType* is only specified for *Variables*; "[<number>]" indicates a single-dimensional array, for multi-dimensional arrays the expression is repeated for each dimension (e.g. [2][3] for a two-dimensional array). For all arrays the *ArrayDimensions* is set as identified by <number> values. If no <number> is set, the corresponding dimension is set to 0, indicating an unknown size. If no number is provided at all the *ArrayDimensions* can be omitted. If no brackets are provided, it identifies a scalar *DataType* and the *ValueRank* is set to the corresponding value (see IEC 62541-3:2015). In addition, *ArrayDimensions* is set to null or is omitted. If it can be Any or ScalarOrOneDimension, the value is put into "{<value>}", so either "{Any}" or "{ScalarOrOneDimension}" and the *ValueRank* is set to the corresponding value (see IEC 62541-3:2015) and the *ArrayDimensions* is set to null or is omitted. In Table 2 examples are given.

Notation	Data-Type	Value-Rank	Array-Dimensions	Description
Int32	Int32	-1	omitted or NULL	A scalar Int32
Int32[]	Int32	1	omitted or {0}	Single-dimensional array of Int32 with an unknown size
Int32[][]	Int32	2	omitted or {0,0}	Two-dimensional array of Int32 with unknown sizes for both dimensions
Int32[3][]	Int32	2	{3,0}	Two-dimensional array of Int32 with a size of 3 for the first dimension and an unknown size for the second dimension
Int32[5][3]	Int32	2	{5,3}	Two-dimensional array of Int32 with a size of 5 for the first dimension and a size of 3 for the second dimension
Int32{Any}	Int32	-2	omitted or NULL	An Int32 where it is unknown if it is scalar or array with any number of dimensions
Int32{ScalarOrOneDimension}	Int32	-3	omitted or NULL	An Int32 where it is either a single-dimensional array or a scalar

Table 2: Examples of DataTypes

- The TypeDefinition is specified for *Objects* and *Variables*.
- The TypeDefinition column specifies a *NodeId* of a *TypeDefinitionNode*, i.e. the specified *Node* points with a *HasTypeDefinition Reference* to the corresponding *TypeDefinitionNode*. The symbolic name of the *NodeId* is used in the table.
- The *ModellingRule* of the referenced component is provided by specifying the symbolic name of the rule in the *ModellingRule* column. In the *AddressSpace*, the *Node* shall use a *HasModellingRule Reference* to point to the corresponding *ModellingRule Object*.

If the *NodeId* of a *DataType* is provided, the symbolic name of the *Node* representing the *DataType* shall be used.

Nodes of all other *NodeClasses* cannot be defined in the same table; therefore only the used *ReferenceType*, their *NodeClass* and their *BrowseName* are specified. A reference to another of this document points to their definition.

If no components are provided, the *DataType*, *TypeDefinition* and *ModellingRule* columns may be omitted and only a *Comment* column is introduced to point to the *Node* definition.

Components of *Nodes* can be complex, i.e. containing components by themselves. The *TypeDefinition*, *NodeClass*, *DataType* and *ModellingRule* can be derived from the type definitions, and the symbolic name can be created as defined in 2.3.3.1. Therefore those containing components are not explicitly specified; they are implicitly specified by the type definitions.

2.3.3 NodeIds and BrowseNames

2.3.3.1 NodeIds

The *NodeIds* of all *Nodes* described in this document are only symbolic names.

The symbolic name of each *Node* defined in this document is its *BrowseName*, or, when it is part of another *Node*, the *BrowseName* of the other *Node*, a “.”, and the *BrowseName* of itself. In this case “part of” means that the whole has a *HasProperty* or *HasComponent Reference* to its part. Since all *Nodes* not being part of another *Node* have a unique name in this document, the symbolic name is unique.

The namespace for this specification is defined in the following chapters. The *NamespaceIndex* for all *NodeIds* defined in this specification is server specific and depends on the position of the namespace URI in the server namespace table.

Note: This specification does not only define concrete *Nodes*, but also requires that some *Nodes* have to be generated, for example one for each device type available in the frame application. The *NodeIds* of those *Nodes* are server-specific, including the *Namespace*. But the *NamespaceIndex* of those *Nodes* cannot be the *NamespaceIndex* used for the *Nodes* defined by this specification, because they are not defined by this specification but generated by the *Server*.

2.3.3.2 BrowseNames

The text part of the *BrowseNames* for all *Nodes* defined in this specification is specified in the tables defining the *Nodes*. The *NamespaceIndex* for all *BrowseNames* defined in this specification is server specific and depends on the position of the namespace URI defined in this specification in the server namespace table.

If the *BrowseName* is not defined by this specification, a namespace index prefix like ‘0:EngineeringUnits’ is added to the *BrowseName*. This is typically necessary if a *Property* of another specification is overwritten or used in the OPC UA types defined in this specification.

2.3.4 Common Attributes

2.3.4.1 General

For all *Nodes* specified in this specification, the *Attributes* named in Table 3 shall be set as specified in the table.

Attribute	Value
DisplayName	The <i>DisplayName</i> is a <i>LocalizedText</i> . Each server shall provide the <i>DisplayName</i> identical to the <i>BrowseName</i> of the <i>Node</i> for the LocaleId "en". Whether the server provides translated names for other LocaleIds is vendor specific.
Description	Optionally a vendor specific description is provided
NodeClass	Shall reflect the <i>NodeClass</i> of the <i>Node</i>
NodeId	The <i>NodeId</i> is described by <i>BrowseNames</i> as defined in 2.3.3.1.
WriteMask	Optionally the <i>WriteMask Attribute</i> can be provided. If the <i>WriteMask Attribute</i> is provided, it shall set all <i>Attributes</i> to not writeable that are not said to be vendor-specific. For example, the <i>Description Attribute</i> may be set to writeable since a Server may provide a server-specific description for the <i>Node</i> . The <i>NodeId</i> shall not be writeable, because it is defined for each <i>Node</i> in this specification.
UserWriteMask	Optionally the <i>UserWriteMask Attribute</i> can be provided. The same rules as for the <i>WriteMask Attribute</i> apply.

Table 3: Common Node Attributes

2.3.4.2 Objects

For all *Objects* specified in this specification, the *Attributes* named in Table 4 shall be set as specified in the table.

Attribute	Value
EventNotifier	Whether the <i>Node</i> can be used to subscribe to <i>Events</i> or not is vendor specific

Table 4: Common Object Attributes

2.3.4.3 Variables

For all *Variables* specified in this specification, the *Attributes* named in Table 5 shall be set as specified in the table.

Attribute	Value
MinimumSamplingInterval	Optionally, a vendor-specific minimum sampling interval is provided
AccessLevel	The access level for <i>Variables</i> used for type definitions is vendor-specific, for all other <i>Variables</i> defined in this part, the access level shall allow a current read; other settings are vendor specific.
UserAccessLevel	The value for the <i>UserAccessLevel Attribute</i> is vendor-specific. It is assumed that all <i>Variables</i> can be accessed by at least one user.
Value	For <i>Variables</i> used as <i>InstanceDeclarations</i> , the value is vendor-specific; otherwise it shall represent the value described in the text.
ArrayDimensions	If the <i>ValueRank</i> does not identify an array of a specific dimension (i.e. <i>ValueRank</i> <= 0) the <i>ArrayDimensions</i> can either be set to null or the <i>Attribute</i> is missing. This behaviour is vendor-specific. If the <i>ValueRank</i> specifies an array of a specific dimension (i.e. <i>ValueRank</i> > 0) then the <i>ArrayDimensions Attribute</i> shall be specified in the table defining the <i>Variable</i> .

Table 5: Common Variable Attributes

2.3.4.4 VariableTypes

For all *VariableTypes* specified in this specification, the *Attributes* named in Table 6 shall be set as specified in the table.

Attributes	Value
Value	Optionally a vendor-specific default value can be provided
ArrayDimensions	If the <i>ValueRank</i> does not identify an array of a specific dimension (i.e. <i>ValueRank</i> \leq 0) the <i>ArrayDimensions</i> can either be set to null or the <i>Attribute</i> is missing. This behaviour is vendor-specific. If the <i>ValueRank</i> specifies an array of a specific dimension (i.e. <i>ValueRank</i> $>$ 0) then the <i>ArrayDimensions</i> <i>Attribute</i> shall be specified in the table defining the <i>VariableType</i> .

Table 6: Common *VariableType* Attributes

3 General information to AutomationML and OPC UA

3.1 Introduction to AutomationML

The AutomationML data format, developed by AutomationML e.V., standardised in IEC 62714, is an open, neutral, XML-based, and free data exchange format which enables a domain and company spanning transfer of engineering data of production systems in a heterogeneous engineering tool landscape. The goal of AutomationML is to interconnect engineering tools in their different disciplines, e.g. plant planning, mechanical engineering, electrical engineering, process engineering, process control engineering, HMI development, PLC programming, robot programming.

AutomationML 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, logic, and behaviour.

In addition, AutomationML follows a modular structure. AutomationML is based on IEC 62424 (CAEX) and integrates other already existing XML-based data formats (see Figure 1). These data formats are used on an “as-is” basis within their own specifications and are not branched for AutomationML needs.

Logically AutomationML is partitioned in:

- description of the plant structure and communication systems expressed as a hierarchy of AutomationML objects and described by means of CAEX following IEC 62424
- description of geometry and kinematics of the different AutomationML objects represented by means of COLLADA 1.4.1 and 1.5.0 (ISO/PAS 17506:2012)
- description of control related logic data of the different AutomationML objects represented by means of PLCopen XML 2.0 and 2.0.1 (IEC61131-10)
- description of relations among AutomationML objects and references to information that is stored in documents outside the top level format

CAEX enables an object oriented approach (see Figure 2) where

- semantics of system objects can be specified using roles defined and collected in role class libraries,
- semantics of interfaces between system objects can be specified using interface classes defined and collected in interface class libraries,
- classes of system objects can be specified using system unit classes (SUC) defined and collected in system unit class libraries, and
- individual objects are modelled in an instance hierarchy (IH) as a hierarchy of internal elements (IE) referencing system unit classes they are derived from, role classes defining its semantics, and interface objects used to interlink objects among each other or with externally modelled information (e.g. COLLADA and PLCopenXML files).

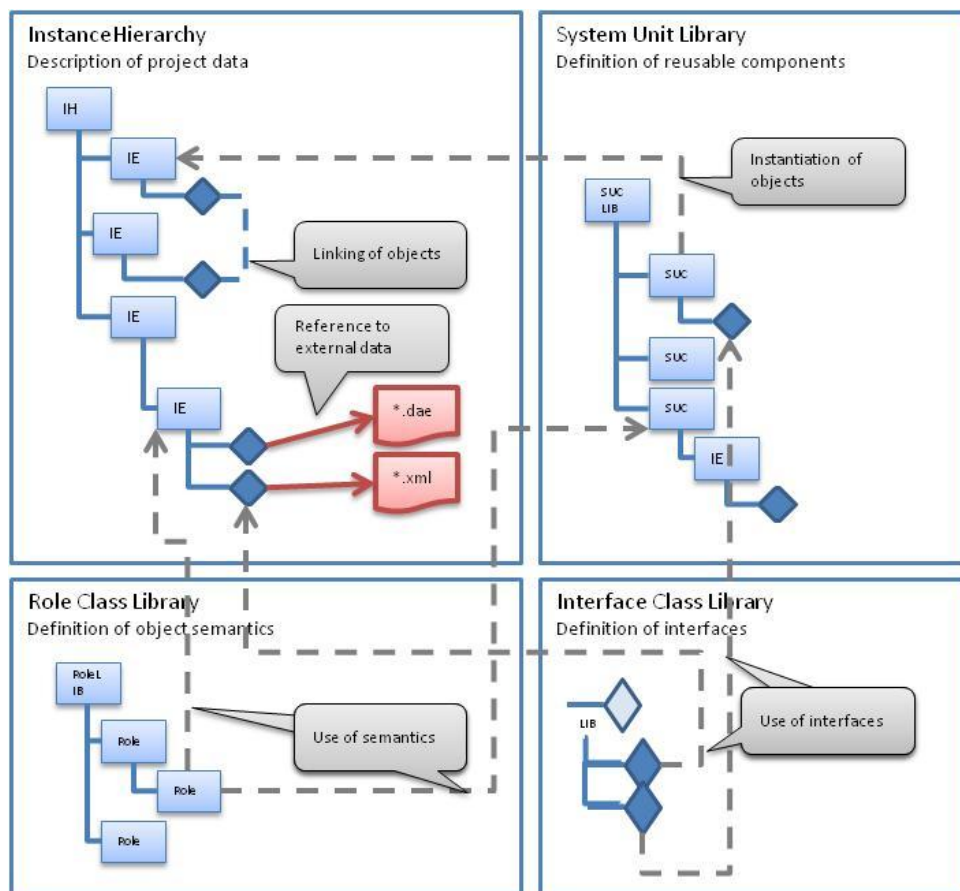


Figure 2: AutomationML topology description architecture

3.2 Introduction to OPC Unified Architecture

3.2.1 General

The main use case for OPC standards is the online data exchange between devices and HMI or SCADA systems using Data Access functionality. In this use case the device data is provided by an OPC server and is consumed by an OPC client integrated into the HMI or SCADA system. OPC DA provides functionality to browse through a hierarchical namespaces containing data items and to read, write and to monitor these items for data changes. The classic OPC standards are based on Microsoft COM/DCOM technology for the communication between software components from different vendors. Therefore classic OPC server and clients are restricted to Windows PC based automation systems.

OPC UA incorporates all features of classic OPC standards like OPC DA, A&E and HDA but defines platform independent communication mechanisms and generic, extensible and object-oriented modelling capabilities for the information a system wants to expose.

The OPC UA network communication part defines different mechanisms optimized for different use cases. The first version of OPC UA is defining an optimized binary TCP protocol for high performance intranet communication as well as a mapping to accepted internet standards like Web Services. The abstract communication model does not depend on a specific protocol mapping and allows adding new protocols in the future. Features like security, access control and reliability are directly built into the transport mechanisms. Based on the platform independence of the protocols, OPC UA servers and clients can be directly integrated into devices and controllers.

The OPC UA *Information Model* provides a standard way for *Servers* to expose *Objects* to *Clients*. *Objects* in OPC UA terms are composed of other *Objects*, *Variables* and *Methods*. OPC UA also allows relationships to other *Objects* to be expressed.

The set of *Objects* and related information that an OPC UA *Server* makes available to *Clients* is referred to as its *AddressSpace*. The elements of the OPC UA *Object Model* are represented in the *AddressSpace* as a set of *Nodes* described by *Attributes* and interconnected by *References*. OPC UA defines eight classes of *Nodes* to represent *AddressSpace* components. The classes are *Object*, *Variable*, *Method*, *ObjectType*, *DataType*, *ReferenceType* and *View*. Each *NodeClass* has a defined set of *Attributes*.

This specification makes use of three essential OPC UA *NodeClasses*: *Objects*, *Methods* and *Variables*.

Objects are used to represent components of a system. An *Object* is associated to a corresponding *ObjectType* that provides definitions for that *Object*.

Methods are used to represent commands or services of a system.

Variables are used to represent values. Two categories of *Variables* are defined, *Properties* and *DataVariables*.

Properties are *Server*-defined characteristics of *Objects*, *DataVariables* and other *Nodes*. *Properties* are not allowed to have *Properties* defined for them. An example for *Properties* of *Objects* is the *Object_Identifier Property* of a *XXX ObjectType*.

DataVariables represent the contents of an *Object*. *DataVariables* may have component *DataVariables*. This is typically used by *Servers* to expose individual elements of arrays and structures. This specification uses *DataVariables* to represent data like the *XXX* of a *XXX Object*.

3.2.2 Graphical Notation

OPC UA defines a graphical notation for an OPC UA *AddressSpace*. It defines graphical symbols for all *NodeClasses* and how different types of *References* between *Nodes* can be visualized. Figure 3 shows the symbols for the six *NodeClasses* used in this specification. *NodeClasses* representing types always have a shadow.

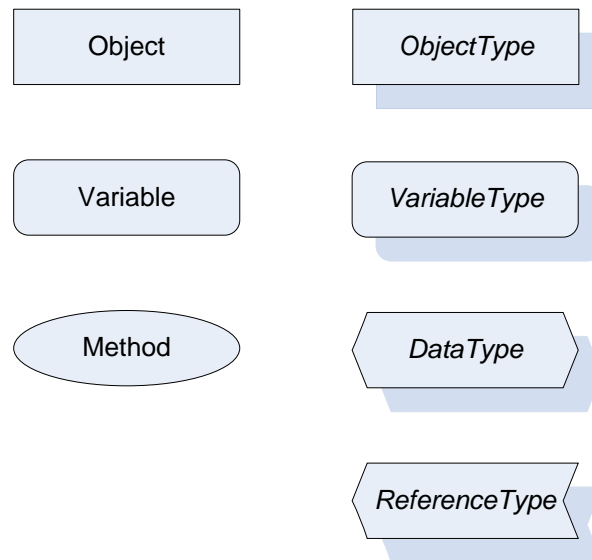


Figure 3: OPC UA Graphical Notation for NodeClasses

Figure 4 shows the symbols for the *ReferenceTypes* used in this specification. The *Reference* symbol is normally pointing from the source *Node* to the target *Node*. The only exception is the *HasSubType* *Reference*. The most important *References* like *HasComponent*, *HasProperty*, *HasTypeDefinition* and *HasSubType* have special symbols avoiding the name of the *Reference*. For other *ReferenceTypes* or derived *ReferenceTypes* the name of the *ReferenceType* is used together with the symbol.

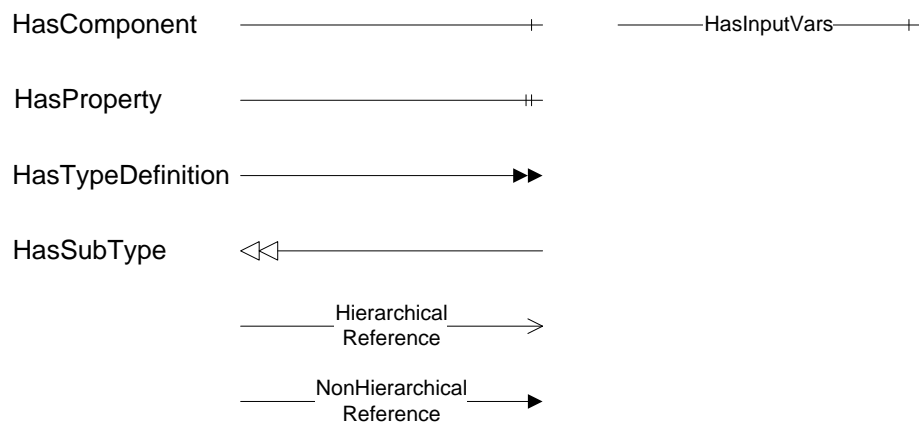


Figure 4: OPC UA Graphical Notation for References

Figure 5 shows a typical example for the use of the graphical notation. *Object_A* and *Object_B* are instances of the *ObjectType_Y* indicated by the *HasTypeDefinition* *References*. The *ObjectType_Y* is derived from *ObjectType_X* indicated by the *HasSubType* *Reference*. The *Object_A* has the components *Variable_1*, *Variable_2* and *Method_1*.

To describe the components of an *Object* on the *ObjectType* the same *NodeClasses* and *References* are used on the *Object* and on the *ObjectType* like for *ObjectType_Y* in the

example. The instance *Nodes* used to describe an *ObjectType* are instance declaration *Nodes*.

To provide more detailed information for a *Node*, a subset or all *Attributes* and their values can be added to a graphical symbol.

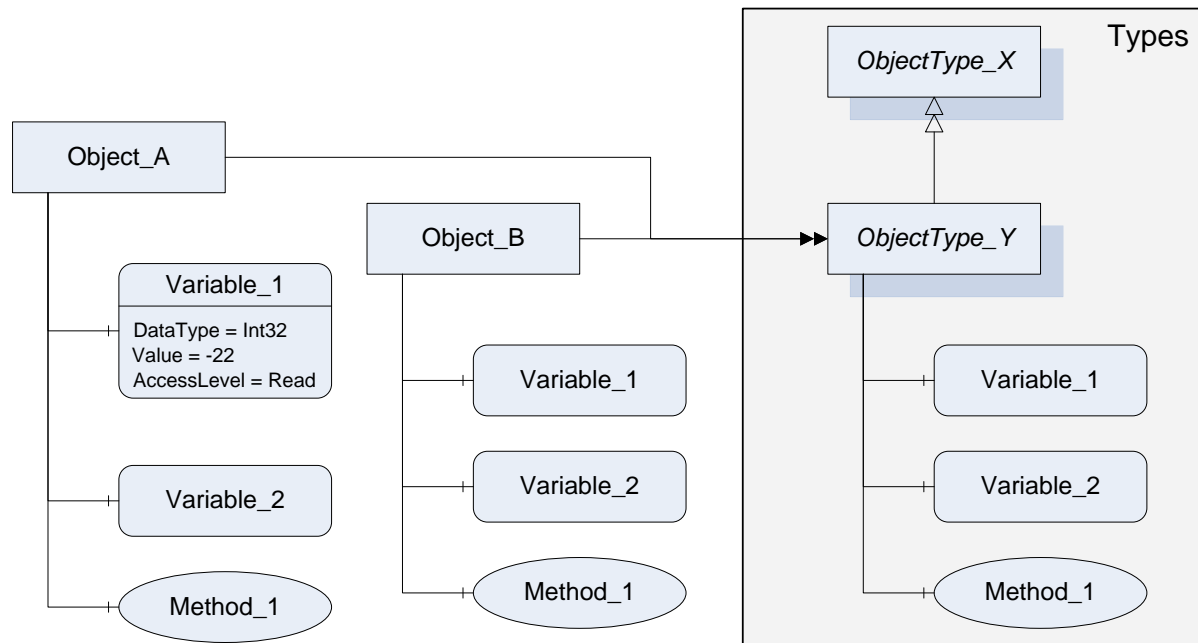


Figure 5: OPC UA Graphical Notation Example

3.3 Use Cases

The specification describes the combination of AutomationML with OPC UA online information like process data and diagnostic information. It extends the application domain of OPC UA (see Figure 6). Therefore, different use cases which will be possible by the combination of both standards were identified.

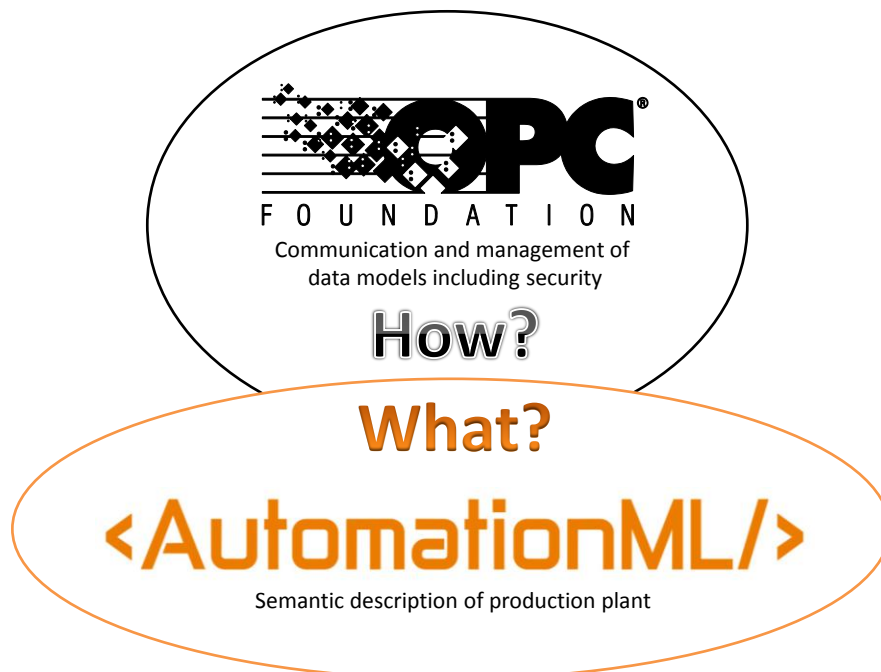


Figure 6: Goals of AutomationML and OPC UA

One opportunity from combining AutomationML and OPC UA is to communicate and operationalise AutomationML by means of OPC UA. It is possible to simplify the creation of OPC UA information models based on existing AutomationML data. This can be realized by a so called OPC UA companion specification due to analogies between AutomationML and the OPC UA information model. The companion specification for AutomationML consists of an object model including many specific semantics which can be used online with multiple involved parties by OPC UA. This makes an online version of the AutomationML model possible - AutomationML models can be exchanged via OPC UA – and includes OPC UA data management, online communication functionality, multi-user support, access methods, security, etc. This is especially important for re-engineering and maintenance use cases where the AutomationML model evolves over time. The present AutomationML model can be managed by OPC UA and makes an up-to-date description of the system as-is possible.

One other opportunity is the lossless exchange of the OPC UA system configuration within AutomationML models. The manual exchange of OPC UA server configuration data will be replaced by a specified description in AutomationML. Parameters to set up OPC UA communication between tools can be exchanged using AutomationML. This realizes consistent data, produces less errors and results in an easier and faster configuration of UA servers and clients. OPC UA can benefit from the description of the complete communication network configuration and structure including communication components of sensors and actuators with respect to communication system parameters, network structure and wiring, quality of service.

4 AutomationML Model Overview

4.1 Modeling concepts

AutomationML models are divided into different hierarchical trees (see Figure 7): InstanceHierarchy, RoleClassLibs, SystemUnitClassLib, and InterfaceClassLib.

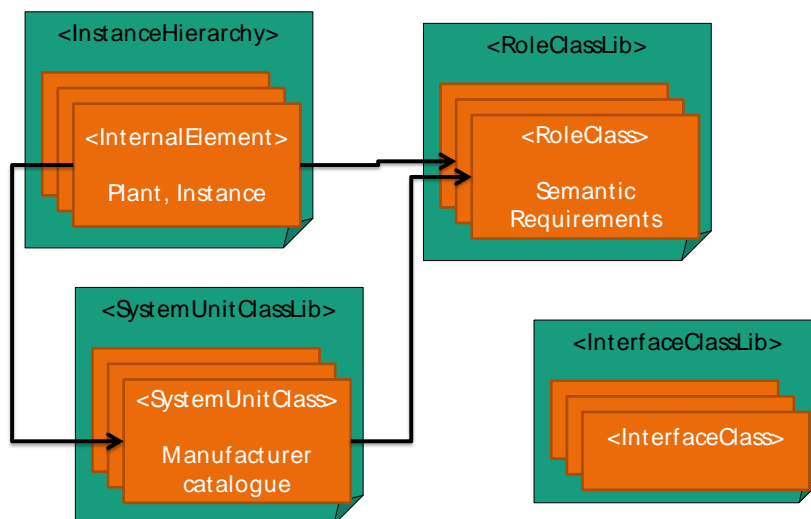


Figure 7: AutomationML hierarchical trees

The main AutomationML elements and all possible relations between them are shown in Figure 8. The organization in a hierarchical tree involves that every element type can have child elements of the same type (see ChildElement relations). SystemUnitClasses and RoleClasses 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 8 describe the possible relations between InternalElements and SystemUnitClasses. An InternalElement can inherit from a SystemUnitClass. In the scope of AutomationML 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 to specify an object in an abstract way and independent of the manufacturer. The assignment of a RoleClass to an object (SystemUnitClass or InternalElement) results in the allocation of fundamental functions or requirements.

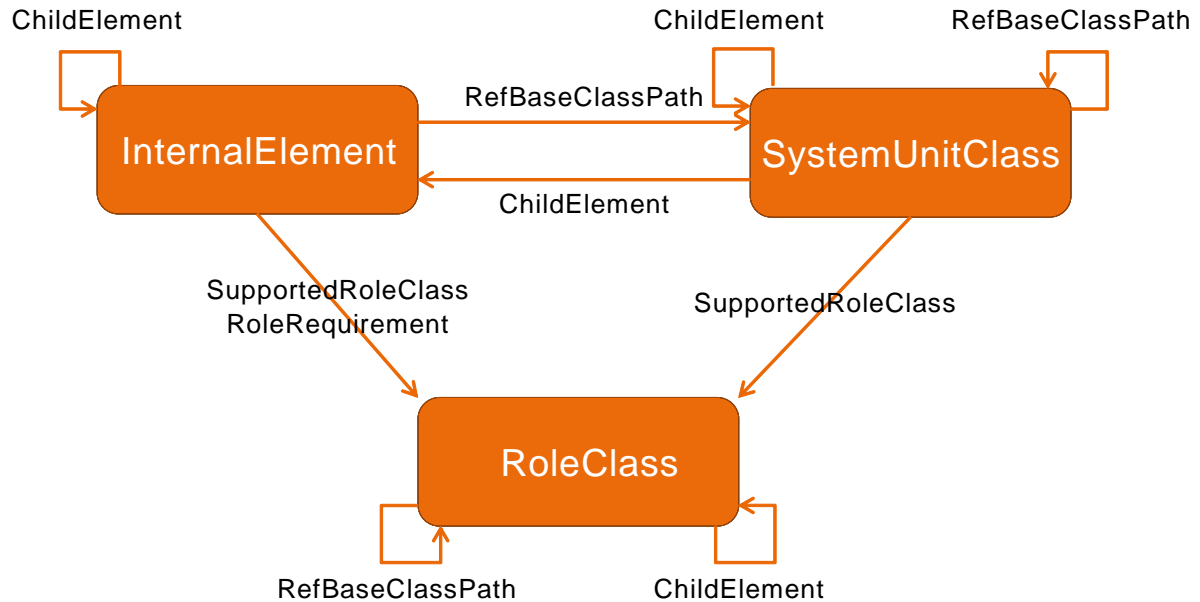


Figure 8: AutomationML main elements

Figure 9 depicts the basic structure of a RoleClass (RC), SystemUnitClass (SUC), or InternalElement (IE). Each can contain arbitrary nested Attributes and Interfaces (see ChildElement relations). The Interfaces are called ExternalInterfaces and shall inherit from an InterfaceClass. They can be connected to other ExternalElements via InternalLinks. The InterfaceClasses are stored hierarchically and shall have independent inheritance relations. InterfaceClasses and certainly ExternalInterfaces may also have arbitrary nested Attributes.

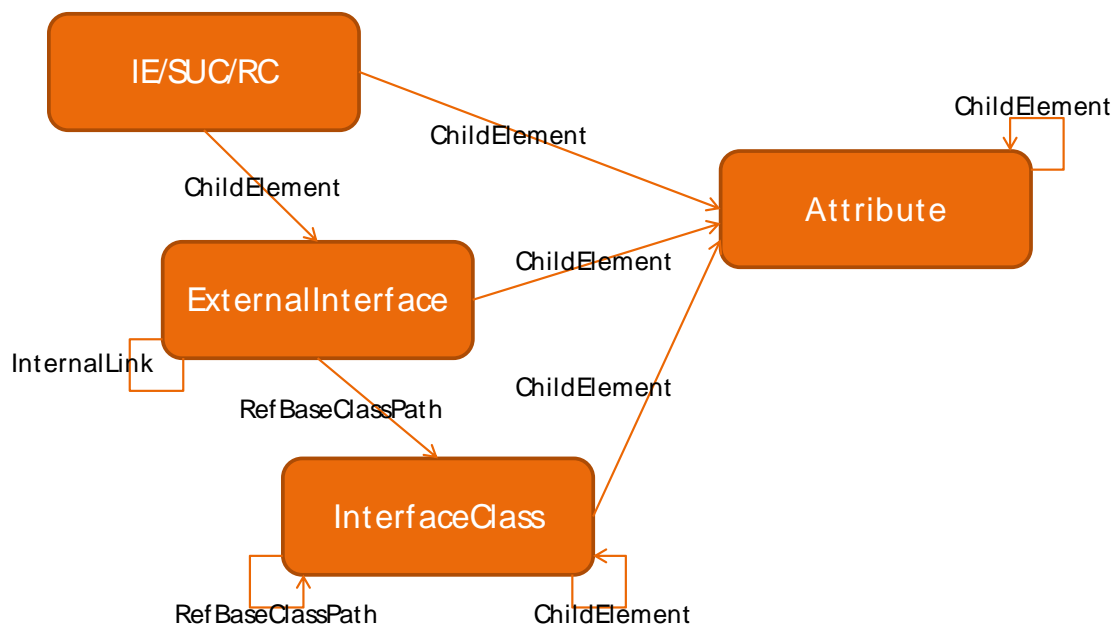


Figure 9: AutomationML main object details.

4.2 Model Overview

4.2.1 Mapping Explanations for Namespaces

Several nodesets (see Figure 10) are defined to organise the address space. Figure 10 includes the nodeset name, the corresponding namespace URI and the file name containing the nodeset. All nodesets are based on the UA Base types. The nodeset AutomationML Base types (AML, see Chapter 5) defines all nodes and type definitions necessary for the modelling of AutomationML in OPC UA. This includes e.g. organisational nodes for the entry into the AutomationML folder for RoleClassLibs or AutomationML types (ObjectType CAEXFileType). One nodeset is defined for the standardised and informative AutomationML libraries (AMLLibs, see Chapter 5.4) coming from IEC°62714-1, IEC62714-2, IEC°62714-3, and IEC°62714-4. And finally, one or more nodesets are used for the concrete AutomationML documents (Filename of each AutomationML document).

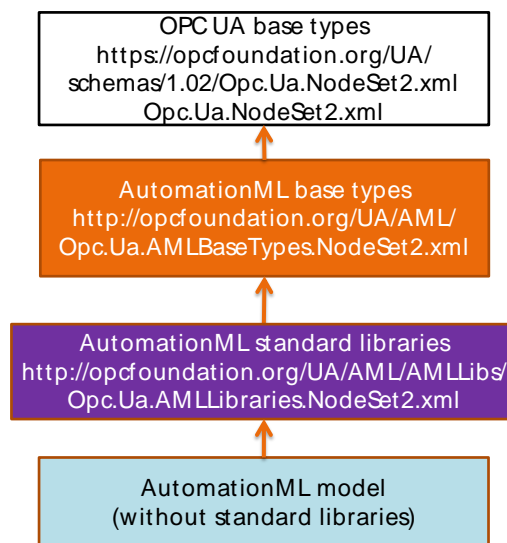


Figure 10: Different nodesets for the usage of AutomationML in OPC UA information models

4.2.2 Mapping Explanations for AutomationML model elements

The main structure of an AutomationML file from Figure 7 is mapped into a slightly variant structure depicted in Figure 11. The InstanceHierarchies and libraries shall be stored in common folders (AutomationMLInstanceHierarchies, AutomationMLLibraries, AutomationMLSystemUnitClassLibs, AutomationMLRoleClassLibs, AutomationMLInterfaceClassLibs).

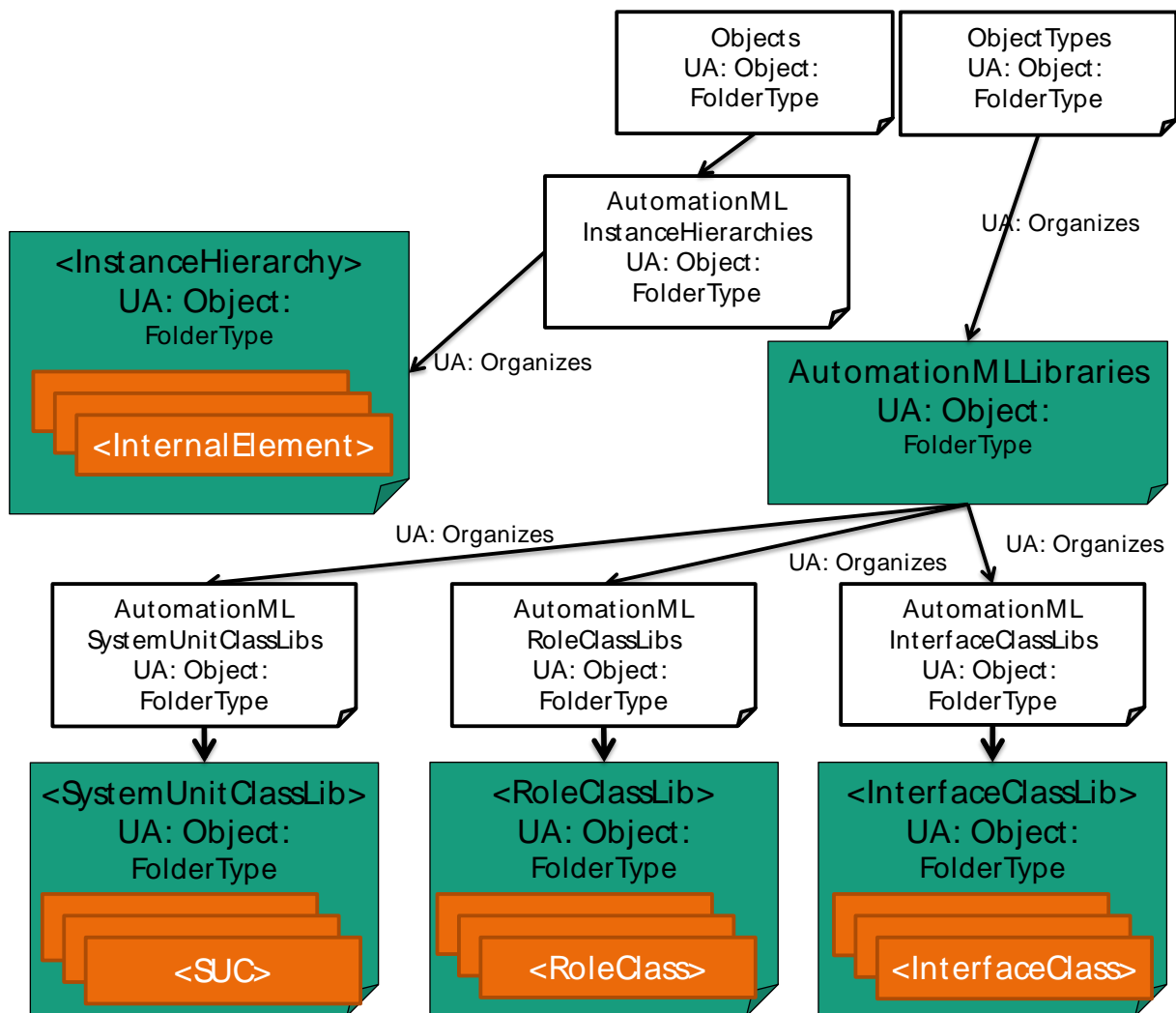


Figure 11: CAEX file <-> OPC UA Types [1]

The AutomationML BaseElementTypes transformation into OPC UA Types is depicted in Figure 12, the transformation of AutomationML elements into OPC UA Types is depicted in Figure 13.

Figure 12 depicts the mapping of the relations between the main AutomationML elements (see Figure 8). The structure of the picture is not changed, so the mapping can be read directly from the graphic. Table 7 and Table 8 resume the mapping rules in a list. Basically, AutomationML InternalElements are represented by OPC UA Objects, AutomationML SystemUnitClasses and RoleClasses are represented by OPC UA ObjectTypes. Detailed definition of new ObjectTypes, ReferenceTypes and VariableTypes are given in Chapter 5.

ChildElement relations within class structures (SystemUnitClass, RoleClass) are mapped to the OPC UA 'Organizes' references. InternalElements which follow the ChildElement relation are referenced via 'HasComponent' references. The inheritance relations within SystemUnitClasses and RoleClasses are mapped to OPC UA 'HasSubtype' references. The internal structures of SystemUnitClasses are depicted by InternalElements which are connected via OPC UA 'HasComponent' references. The RefBaseClassPath relation of AutomationML is an OPC UA 'HasTypeDefinition' reference. The referencing of RoleClasses to InternalElements or SystemUnitClasses via SupportedRoleClass or RoleRequirement of AutomationML is done via OPC UA 'HasAMLRoleReference' references.

AutomationML	OPC UA
InternalElement	Object
SystemUnitClass	ObjectType
RoleClass	ObjectType

Table 7: Element mapping

AutomationML Source	AutomationML Target	AutomationML Relation	OPC UA Reference	Description
InternalElement	RoleClass	SupportedRoleClass	HasAMLRoleReference	Semantic similar to HasTypeDefinition
InternalElement	RoleClass	RoleRequirement	HasAMLRoleReference	Semantic similar to HasTypeDefinition
InternalElement	SystemUnitClass	RefBaseClassPath	HasTypeDefinition	
InternalElement	InternalElement		HasComponent	ChildElement
SystemUnitClass	RoleClass	SupportedRoleClass	HasAMLRoleReference	Semantic similar to HasTypeDefinition
SystemUnitClass	InternalElement		HasComponent	ChildElement
SystemUnitClass	SystemUnitClass		Organizes	ChildElement
SystemUnitClass	SystemUnitClass	RefBaseClassPath	HasSubtype	
RoleClass	RoleClass		Organizes	ChildElement
RoleClass	RoleClass	RefBaseClassPath	HasSubtype	

Table 8: Relation mapping

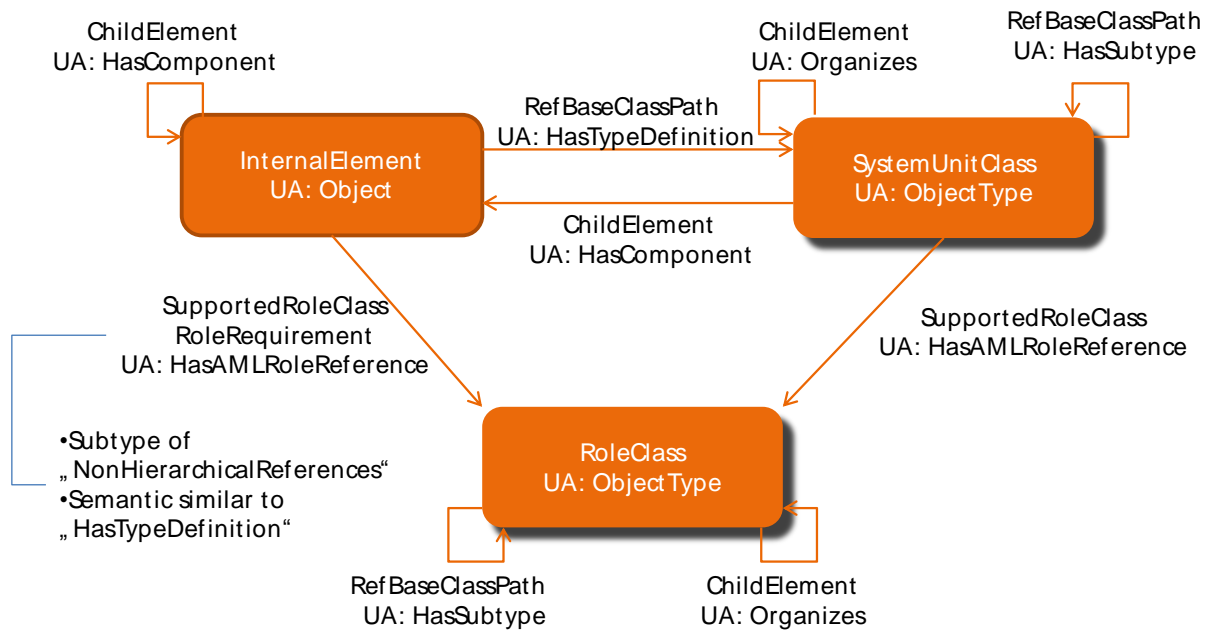


Figure 12: AutomationML BaseElementTypes and the OPC UA Types [1]

The mappings of the AutomationML object details (see Figure 9) are depicted in Figure 13. The analogy between the two figures is obvious. Table 9 and Table 10 summarize the mapping rules in a list. AutomationML Attributes become OPC UA Variables. AutomationML InterfaceClasses are transformed into OPC UA ObjectTypes and the corresponding AutomationML ExternalInterface elements are modelled by OPC UA 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. Similarly to the RoleClasses and SystemUnitClasses, the relations between InterfaceClasses are mapped to 'Organizes' references within the XML tree structure and to 'HasSubtype' references concerning inheritance relation. Relations between ExternalInterfaces (AutomationML InternalLinks) are assigned via 'HasAMLInternalLink' reference. Details are given in Chapter 5.

AutomationML	OPC UA
InternalElement	Object
SystemUnitClass	ObjectType
RoleClass	ObjectType
InterfaceClass	ObjectType
Attribute	Variable
ExternalInterface	Object

Table 9: Element mapping

AutomationML Source	AutomationML Target	AutomationML Relation	OPC UA Reference	Description
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

Table 10: Relation mapping

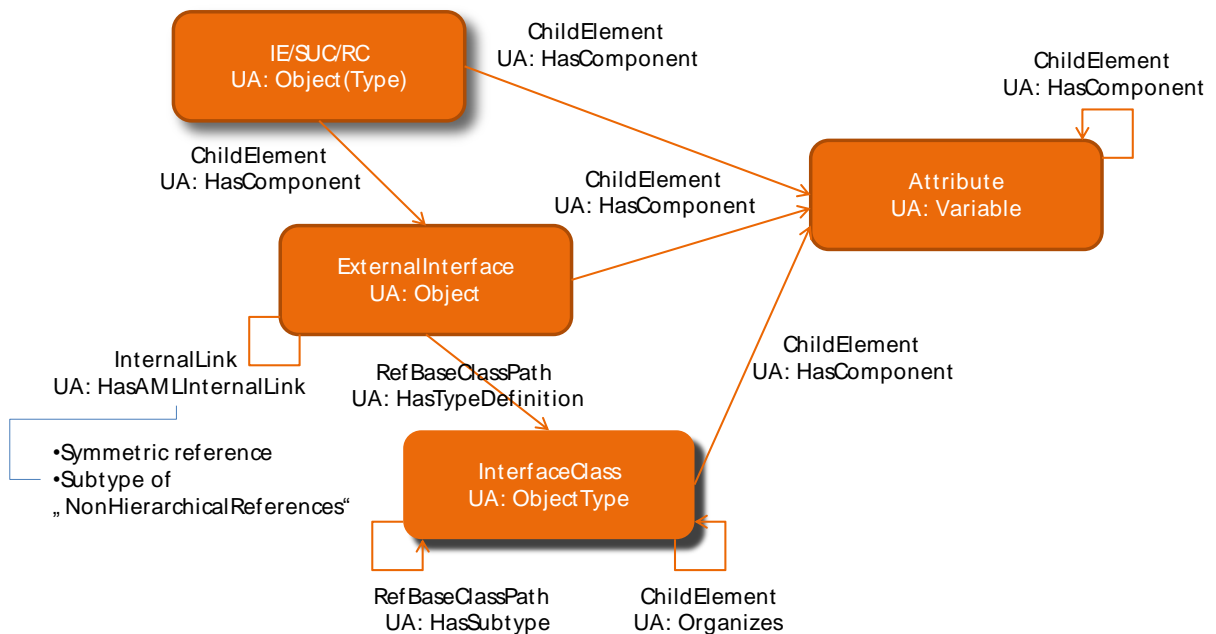


Figure 13: AutomationML Element and the OPC UA Types [1]

4.3 Mapping Example

Figure 14 depicts the tree structure of an AutomationML example. This example shows a simple AutomationML document (version 2.0) which contains:

- one InstanceHierarchy with InternalElements
- one SystemUnitClassLib with one SystemUnitClass
- one RoleClassLib with one RoleClass and
- one InterfaceClassLib with one InterfaceClass.

The elements in this example are designed to model some aspects of a manufacturing system. The RoleClassLib contains a RoleClass to characterize a system component as a 'Tool'. The InterfaceClassLib contains an InterfaceClass which models the energy supply of any tool. The SystemUnitClassLib contains a class representing an electric screwdriver. The InstanceHierarchy describes a manufacturing system containing the two screwdrivers in the system.

The used AutomationML-Libraries (AutomationMLBaseRoleClassLibrary and AutomationMLInterfaceClassLibrary) are stored in separated CAEX files and are included via external references.

Annex A contains the XML text of the example in AutomationML.

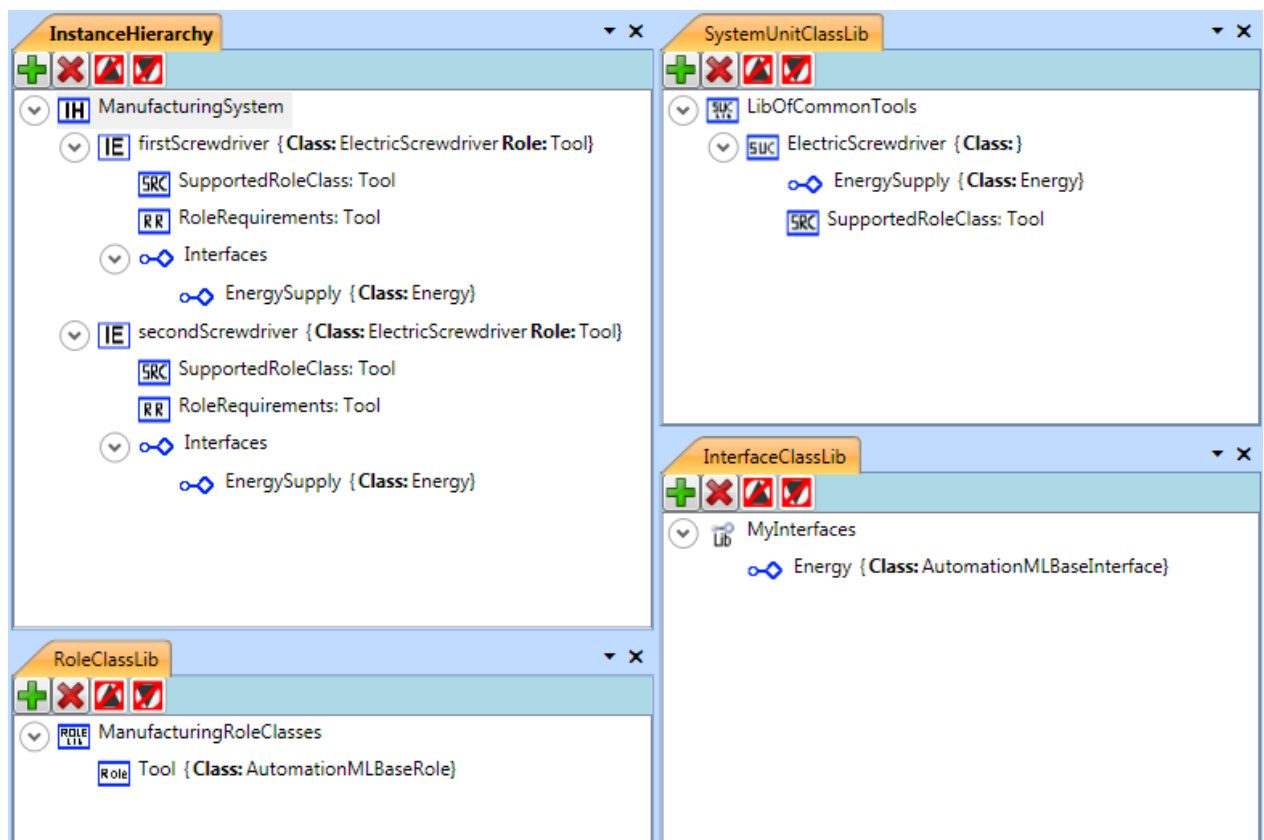


Figure 14: AutomationML example.

The example shown in Figure 14 will now be redefined within OPC UA. Annex A contains the XML text of the example in OPC UA.

Figure 15 depicts the main structure of the example including the folder objects for the organization of the address space. The used graphical notation is defined by the OPC Foundation and is explained in chapter 3.2.2. It includes the OPC UA Objects used to organize the address space structure which are explained in chapter 6.4.1.

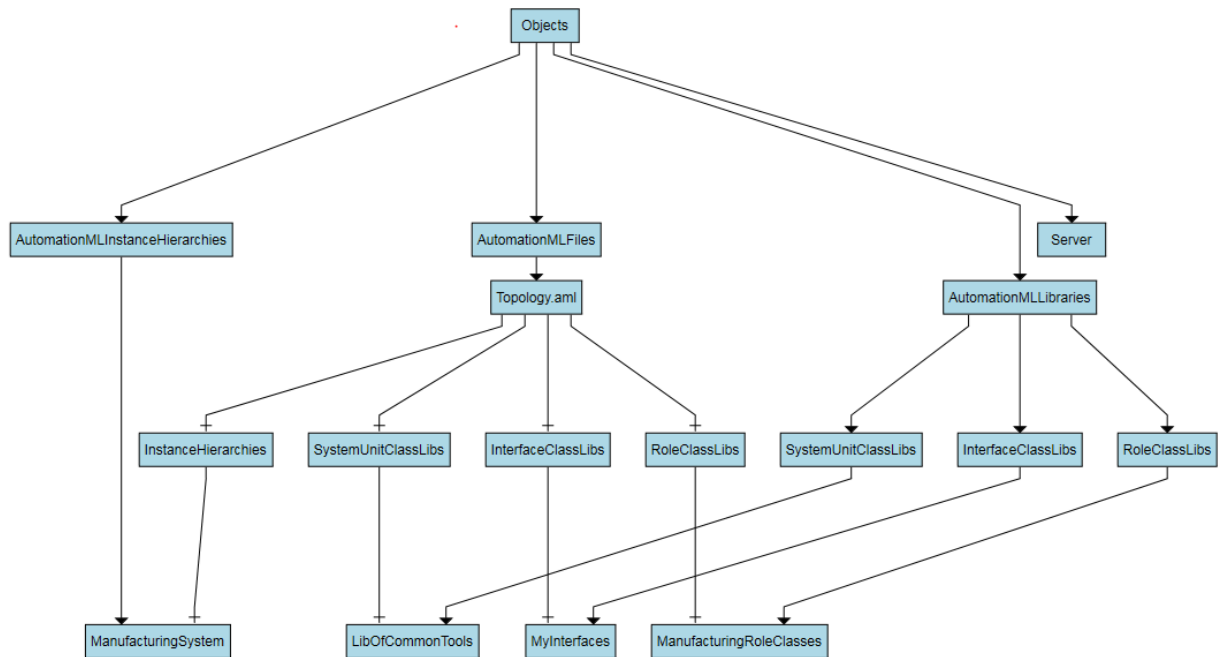


Figure 15: Example main structure in OPC UA.

Figure 16 depicts the role classes of the example. It includes the inheritance structure of the roles. In the RoleClassLibs 'ManufacturingRoleClasses' the RoleClass 'Tool' is included.

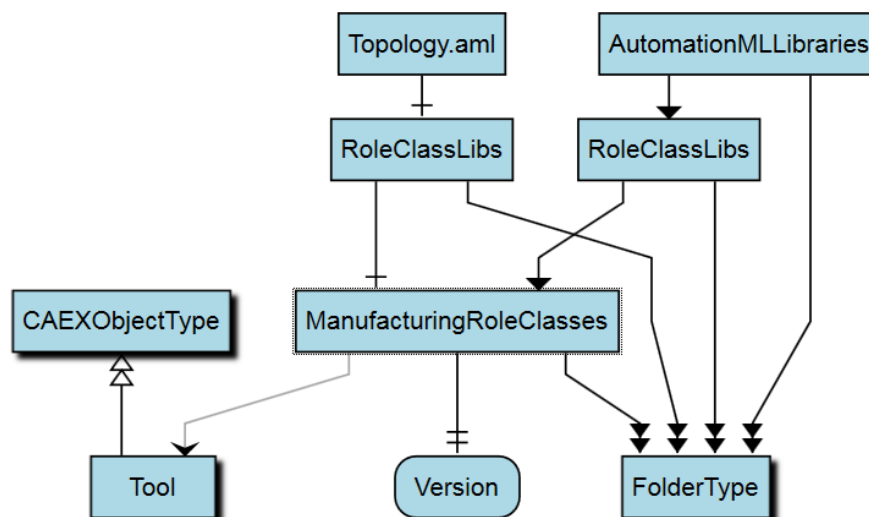


Figure 16: Example role classes in OPC UA.

Figure 17 shows the SystemUnitClasses. The SystemUnitClassLib 'LibOfCommonTools' includes a SystemUnitClass 'ElectricScrewdriver' with reference to the role 'Tool'. This SystemUnitClass includes an ExternalInterface 'EnergySupply'.

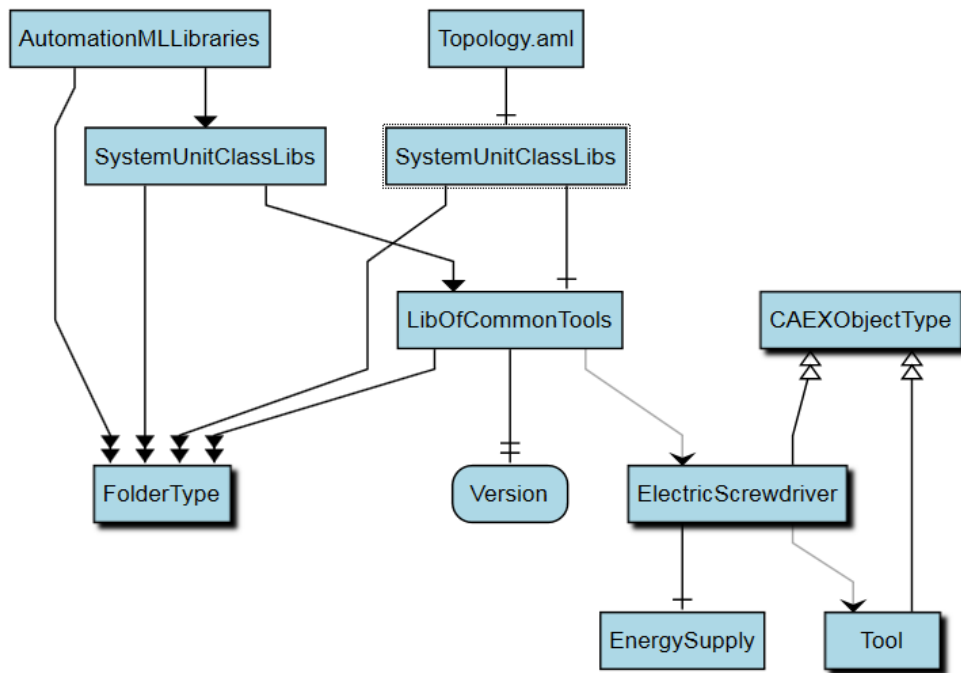


Figure 17: Example system unit classes in OPC UA.

In Figure 18 you can find the InterfaceClassLib 'MyInterfaces' including the InterfaceClass 'Energy'.

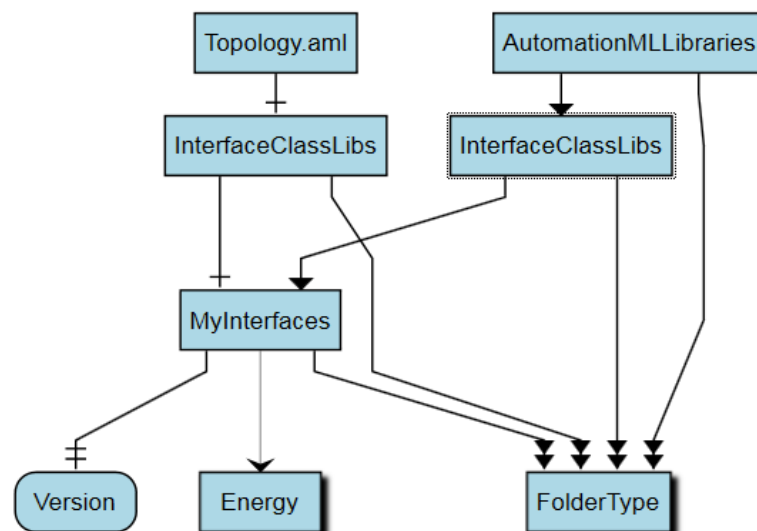


Figure 18: Example interface classes in OPC UA.

The mapping of the InstanceHierarchy is shown in Figure 19. The File 'Topology.aml' includes the InstanceHierarchy 'ManufacturingSystem' which includes two InternalElements 'firstScrewdriver' and 'secondScrewdriver'. They are derived from the SystemUnitClass 'ElectricScrewdriver' and inherit its structure and elements, such as a variable 'ID', an ExternalInterface 'EnergySupply' and the assigned RoleClass 'Tool'. The InternalElement 'firstScrewdriver' additionally consists of a variable 'NewAttribute'.

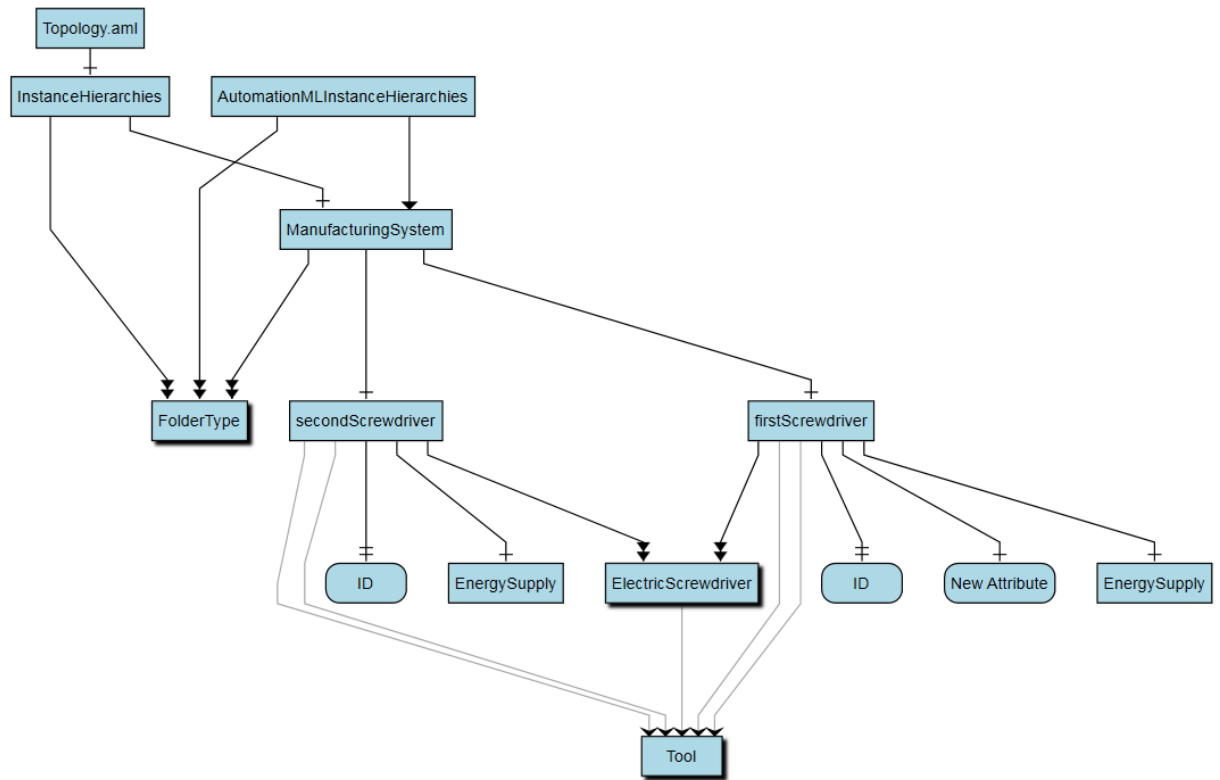


Figure 19: Example InstanceHierarchy in OPC UA.

5 AutomationML Base Types OPC UA Model

5.1 ObjectTypes

5.1.1 General

The property ID of the ObjectType nodes describes a unique identifier.

The property Version of the ObjectType nodes consists of organizational information about the state of the version.

5.1.2 CAEXBasicObjectType

5.1.2.1 General

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

5.1.2.2 ObjectType Definition

The *CAEXBasicObjectType* is formally defined in Table 11.

Attribute	Value				
BrowseName	CAEXBasicObjectType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the BaseObjectType					
HasProperty	Variable	Version	String	PropertyType	Mandatory

Table 11: CAEXBasicObjectType Definition

5.1.2.3 ObjectType Description

5.1.2.3.1 Version

Version provides the version number for the CAEXBasicObjectType

5.1.3 CAEXFileType

5.1.3.1 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.

5.1.3.2 ObjectType Definition

The *CAEXFileType* is formally defined in Table 12.

Attribute	Value				
BrowseName	CAEXFileType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the CAEXBasicObjectType					
HasComponent	Object	InstanceHierarchies		FolderType	Mandatory
HasComponent	Object	InterfaceClassLibs		FolderType	Mandatory
HasComponent	Object	RoleClassLibs		FolderType	Mandatory
HasComponent	Object	SystemUnitClassLibs		FolderType	Mandatory

Table 12: CAEXFileType Definition

5.1.3.3 ObjectType Description

5.1.3.3.1 InstanceHierarchies

The InstanceHierarchies folder includes all CAEX InstanceHierarchies of a CAEX file.

5.1.3.3.2 InterfaceClassLibs

The InterfaceClassLibs folder includes all CAEX InterfaceClassLibs of a CAEX file.

5.1.3.3.3 RoleClassLibs

The RoleClassLibs folder includes all CAEX RoleClassLibs of a CAEX file.

5.1.3.3.4 SystemUnitClassLibs

The SystemUnitClassLibs folder includes all CAEX SystemUnitClassLibs of a CAEX file.

5.1.4 CAEXObjectType

5.1.4.1 General

The CAEXObjectType defines all general characteristics of a CAEX object. All other CAEX objects derive from it. The CAEXObjectType inherits all Properties of the CAEXBasicObjectType.

5.1.4.2 ObjectType Definition

The CAEXObjectType is formally defined in Table 13.

Attribute	Value				
BrowseName	CAEXObjectType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the CAEXBasicObjectType					
HasProperty	Variable	ID	String	PropertyType	Mandatory

Table 13: CAEXObjectType Definition

5.1.4.3 ObjectType Description

5.1.4.3.1 ID

ID provides a unique ID of the CAEXObjectType which shall be in form of a UUID.

5.1.5 AutomationMLBaseInterface

5.1.5.1 General

The AutomationMLBaseInterface defines all general characteristics of a CAEX InterfaceClass object. All other InterfaceClass objects derive from it. The CAEXObjectType inherits all Properties of the CAEXObjectType.

5.1.5.2 ObjectType Definition

The AutomationMLBaseInterface is formally defined in Table 14.

Attribute	Value				
BrowseName	AutomationMLBaseInterface				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the CAEXObjectType					

Table 14: AutomationMLBaseInterface Definition

5.1.6 AutomationMLBaseRole

5.1.6.1 General

The AutomationMLBaseRole defines all general characteristics of a CAEX RoleClass object. All other RoleClass objects derive from it. The AutomationMLBaseRole inherits all Properties of the CAEXObjectType.

5.1.6.2 ObjectType Definition

The AutomationMLBaseRole is formally defined in Table 15.

Attribute	Value				
BrowseName	AutomationMLBaseRole				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the CAEXObjectType					

Table 15: AutomationMLBaseRole Definition

5.1.7 AutomationMLBaseSystemUnit

5.1.7.1 General

The AutomationMLBaseSystemUnit defines all general characteristics of a CAEX SystemUnitClass object. All other SystemUnitClass objects derive from it. The CAEXObjectType inherits all Properties of the CAEXObjectType.

5.1.7.2 ObjectType Definition

The AutomationMLBaseSystemUnit is formally defined in Table 16.

Attribute	Value				
BrowseName	AutomationMLBaseSystemUnit				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the CAEXObjectType					

Table 16: AutomationMLBaseSystemUnit Definition

5.2 ReferenceTypes

Following the AutomationML specific OPC UA definitions will be listed.

5.2.1 HasAMLRoleReference

5.2.1.1 General

The *HasAMLRoleReference* is a concrete *ReferenceType* that can be used directly. It is a subtype of *NonHierarchicalReference*.

5.2.1.2 ObjectType Definition

The *HasAMLRoleReference* is formally defined in Table 17.

Attribute	Value		
BrowseName	HasAMLRoleReference		
InverseName	IsSupportedRole		
Symmetric	False		
IsAbstract	False		
References	NodeClass	BrowseName	Comment
Subtype of <i>NonHierarchicalReference</i> ReferenceType defined in OPC Part 5			

Table 17: HasAMLRoleReference

This *ReferenceType* is used to describe a SupportedRoleClass or RoleRequirement relation in AutomationML.

The *SourceNode* of this *ReferenceType* shall be an object.

The *TargetNode* of this *ReferenceType* shall be a subtype of AMLRoleClass.

5.2.2 HasAMLInternalLink

5.2.2.1 General

The *HasAMLInternalLink* is a concrete *ReferenceType* that can be used directly. It is a subtype of *NonHierarchicalReference*.

5.2.2.2 ReferenceType Definition

The *HasAMLInternalLink* is formally defined in Table 18.

Attribute	Value		
BrowseName	HasAMLInternalLink		
InverseName	HasAMLInternalLink		
Symmetric	True		
IsAbstract	False		
References	NodeClass	BrowseName	Comment
Subtype of <i>NonHierarchicalReference</i> ReferenceType defined in OPC Part 5			

Table 18: *HasAMLInternalLink*

This *ReferenceType* is used to describe an *InternalLink* relation in AutomationML.

The *SourceNode* of this *ReferenceType* shall be an object which is derived from the *AutomationMLBaseInterface* or one of its subtypes.

The *TargetNode* of this *ReferenceType* shall be an object which is derived from the *AutomationMLBaseInterface* or one of its subtypes.

5.3 VariableTypes

5.3.1 AMLBaseVariableType

5.3.1.1 General

The *AMLBaseVariableType* defines all general characteristics of AutomationML attributes. All other AutomationML attributes derive from it.

5.3.1.2 VariableType Definition

The *AMLBaseVariableType* is formally defined in Table 19.

Attribute	Value				
BrowseName	AMLBaseVariableType				
IsAbstract	False				
ValueRank	-1				
DataType	BaseDataType				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the <i>BaseVariableType</i>					
HasProperty	Variable	ID	String	PropertyType	Optional
HasProperty	Variable	Version	String	PropertyType	Optional

Table 19: *AMLBaseVariableType* Definition

5.3.1.3 VariableType Description

5.3.1.3.1 ID

ID provides a unique ID of the *AMLBaseVariableType* which shall be in form of a UUID.

5.3.1.3.2 Version

Version provides the version number for the AMLBaseVariableType

5.4 Mapping of AutomationML XML DataTypes to OPC UA DataTypes

The mapping of AutomationML XML DataTypes to OPC UA DataTypes is described in Table 20.

XML SimpleTyp	OPC UA DataType	
string	String	
boolean	Boolean	TRUE or FALSE
decimal	Double	
float	Float	
double	Double	
duration	Duration	Double (time in milliseconds)
dateTime	DateTime	64bit signed Int (number of 100 nanosecond intervals since January 1, 1601 (UTC)
time	Duration	
date	DateTime	dd.mm.yyyy 00:00
gYearMonth	DateTime	1.mm.yyyy 00:00
gYear	DateTime or Int16	1.1.yyyy 00:00 or Int16
gMonthDay	Duration or String	
gDay	Duration	
gMonth	Duration or String	
hexBinary	ByteString	Sequence of 0-255
base64Binary	ByteString	
anyURI	String	
QName	String	Prefix->NamespaceIndex
NOTATION	ListOfString	
normalizedString	String	
token	String	
language	LocaleId	
NMTOKEN	String	
NMTOKENS	ListOfString	
Name	String	
NCName	String	
ID	String	
IDREF	String	
IDREFS	ListOfString	
ENTITY	String	
ENTITIES	ListOfString	
integer	Int64	Integer is abstract --> Int64
nonPositiveInteger	Int64	
negativeInteger	Int64	
long	Int64	
int	Int32	
short	Int16	
byte	SByte	
nonNegativeInteger	UInt64	UInteger is abstract --> UInt64
unsignedLong	UInt64	
unsignedInt	UInt32	
unsignedShort	UInt16	
unsignedByte	Byte	
positiveInteger	Int64	

Table 20: Mapping of XML data types to OPC UA BaseTypes

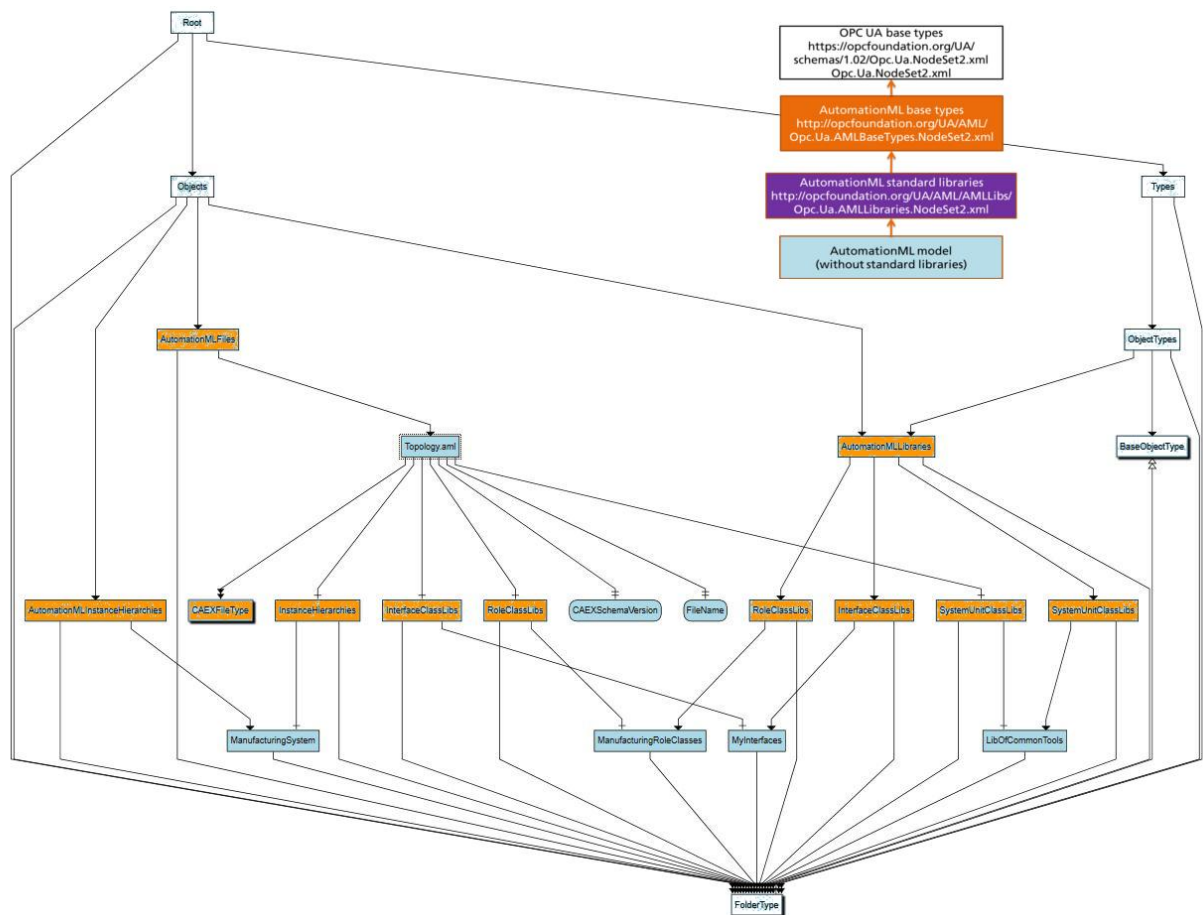
5.4.1 OPC UA Objects used to organize the address space structure

The access to the model is split up into an AutomationML specific entry possibility and an OPC UA specific entry possibility. The organizational nodes facilitate the navigation (browse) to elements. Figure 20 depicts an example address space. Different colours represent different nodesets as explained in chapter 4.2.1.

For users familiar with OPC UA, objects and object types can be accessed via the classical UA way (Objects folder and ObjectTypes folder). The global instance view can be reached via the Objects node and a specific node called AutomationMLInstanceHierarchies of type FolderType. The global type view can be reached via the ObjectTypes node and the corresponding AutomationML-specific object types.

The File_xyz node consists of nodes of type FolderType called

- Below File_xyz all references must be browsable bidirectionally (isForward = False).



The naming for the AutomationML specific nodes is as follows: AutomationMLFiles, AutomationMLLibraries, AutomationMLInstanceHierarchies, CAEXBaseType are entry points into the hierarchies. They include AutomationML or CAEX in their browse names. Everything below these nodes does not need an AutomationML or CAEX prefix anymore, e.g. RoleClassLibs.

44

5.4.1.1 Instances and entry points

5.4.1.1.1 AutomationMLFiles

This Instance is a child of Objects. It is defined in Table 21.

The AutomationMLFiles folder represents a browse entry point when looking for AutomationML files in the OPC UA information model.

Name	TypeDefinition
AutomationMLFiles	FolderType

Table 21: AutomationMLFiles Instance Definition

5.4.1.1.2 AutomationMLInstanceHierarchies

This Instance is a child of Objects. It is defined in Table 22.

The AutomationMLInstanceHierarchies folder represents a browse entry point when looking for AutomationML InstanceHierarchies in the OPC UA information model.

Name	TypeDefinition
AutomationMLInstanceHierarchies	FolderType

Table 22: AutomationMLInstanceHierarchies Instance Definition

5.4.1.1.3 AutomationMLLibraries

This Instance is a child of Objects. It is defined in Table 23.

The AutomationMLLibraries folder represents a browse entry point when looking for AutomationML libraries in the OPC UA information model.

Name	TypeDefinition
AutomationMLLibraries	FolderType

Table 23: AutomationMLLibraries Instance Definition

5.4.1.1.4 InterfaceClassLibs

This Instance is a child of AutomationMLLibraries. It is defined in Table 24.

The InterfaceClassLibs folder represents a browse entry point when looking for AutomationML InterfaceClassLibs in the OPC UA information model.

Name	TypeDefinition
InterfaceClassLibs	FolderType

Table 24: InterfaceClassLibs Instance Definition

5.4.1.1.5 RoleClassLibs

This Instance is a child of AutomationMLLibraries. It is defined in Table 25.

The RoleClassLibs folder represents a browse entry point when looking for AutomationML RoleClassLibs in the OPC UA information model.

Name	TypeDefinition
RoleClassLibs	FolderType

Table 25: RoleClassLibs Instance Definition

5.4.1.1.6 SystemUnitClassLibs

This Instance is a child of AutomationMLLibraries. It is defined in Table 26.

The SystemUnitClassLibsfolder represents a browse entry point when looking for AutomationML SystemUnitClassLibs in the OPC UA information model.

Name	TypeDefinition
SystemUnitClassLibs	FolderType

Table 26: SystemUnitClassLibs Instance Definition

6 AutomationML Libraries OPC UA Model

6.1 General

The property ID of the ObjectType nodes describes a unique identifier.

The property Version of the ObjectType nodes consists of organizational information about the state of the version.

6.2 AutomationMLInterfaceClassLib

6.2.1 Instances

6.2.1.1 AutomationMLInterfaceClassLib

This *Instance* is a child of *InterfaceClassLibs*. It is defined in Table 27.

It consists of the Standard AutomationML Interface Class Library (Part 1). The content is extended with Part 3 and Part 4 of the AutomationML standard specification.

Name	TypeDefinition
AutomationMLInterfaceClassLib	FolderType

Table 27: AutomationMLInterfaceClassLib Instance Definition

6.2.2 ObjectTypes

6.2.2.1 AutomationMLBaseInterface

6.2.2.1.1 General

The interface class “AutomationMLBaseInterface” is a base abstract interface type and shall be used as parent for the description of all AML interface classes.

6.2.2.1.2 ObjectType Definition

The *AutomationMLBaseInterface* is formally defined in Table 28.

Attribute	Value				
BrowseName	AutomationMLBaseInterface				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the CAEXObjectType					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 28: AutomationMLBaseInterface Definition

6.2.2.2 AttachmentInterface

6.2.2.2.1 General

The interface class “AttachmentInterface” shall be used to denote a geometrical connection point of an AutomationML object.

6.2.2.2.2 ObjectType Definition

The *AttachmentInterface* is formally defined in Table 29.

Attribute	Value				
BrowseName	AttachmentInterface				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the AutomationMLBaseInterface					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 29: AttachmentInterface Definition

6.2.2.3 Communication

6.2.2.3.1 General

The interface class “Communication” shall be used for the description of communication related interfaces.

6.2.2.3.2 ObjectType Definition

The *Communication* is formally defined in Table 30.

Attribute	Value				
BrowseName	Communication				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the AutomationMLBaseInterface					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 30: Communication Definition

6.2.2.4 SignalInterface

6.2.2.4.1 General

The interface class “SignalInterface” shall be used for modelling signals.

6.2.2.4.2 ObjectType Definition

The *SignalInterface* is formally defined in Table 31.

Attribute	Value				
BrowseName	SignalInterface				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Communication					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 31: SignalInterface Definition

6.2.2.5 ExternalDataConnector

6.2.2.5.1 General

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.

6.2.2.5.2 ObjectType Definition

The *ExternalDataConnector* is formally defined in Table 32.

Attribute	Value				
BrowseName	ExternalDataConnector				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the AutomationMLBaseInterface					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasComponent	Variable	refURI	String	BaseDataVariableType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 32: ExternalDataConnector Definition

6.2.2.5.3 ObjectType Description

6.2.2.5.3.1 refURI

The attribute “refURI” shall be used in order to store the path to the reference external document.

6.2.2.6 COLLADAInterface

6.2.2.6.1 General

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.

6.2.2.6.2 ObjectType Definition

The *COLLADAInterface* is formally defined in Table 33.

Attribute	Value				
BrowseName	COLLADAInterface				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the ExternalDataConnector					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 33: COLLADAInterface Definition

6.2.2.6.3 ObjectType Description

6.2.2.7 PLCopenXMLInterface

6.2.2.7.1 General

The interface class “PLCopenXMLInterface” shall be used in order to reference external PLCopenXML documents and to publish interfaces that are defined inside an external PLCopenXML document. Details are intended to be specified in IEC 62714-3.

6.2.2.7.2 ObjectType Definition

The *PLCopenXMLInterface* is formally defined in Table 34.

Attribute	Value				
BrowseName	PLCopenXMLInterface				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the ExternalDataConnector					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 34: PLCopenXMLInterface Definition

6.2.2.8 LogicInterface

6.2.2.8.1 General

The interface class “LogicInterface” shall be used in order to reference a logic description within an external PLCopen XML document.

6.2.2.8.2 ObjectType Definition

The *LogicInterface* is formally defined in Table 35.

Attribute	Value				
BrowseName	LogicInterface				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the PLCopenXMLInterface					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 35: LogicInterface Definition

6.2.2.9 VariableInterface

6.2.2.9.1 General

The interface class “VariableInterface” shall be used in order to reference variables in external PLCopen XML document.

6.2.2.9.2 ObjectType Definition

The *VariableInterface* is formally defined in Table 36.

Attribute	Value				
BrowseName	VariableInterface				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the PLCOpenXMLInterface					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 36: *VariableInterface* Definition

6.2.2.10 InterlockingVariableInterface

6.2.2.10.1 General

The interface class “InterlockingVariableInterface” shall be used in order to reference variables representing an interlocking condition in external PLCOpen XML document.

6.2.2.10.2 ObjectType Definition

The *InterlockingVariableInterface* is formally defined in Table 37.

Attribute	Value				
BrowseName	InterlockingVariableInterface				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the VariableInterface					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasComponent	Variable	SafeConditionEquals	Boolean	BaseDataVariableType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 37: *InterlockingVariableInterface* Definition

6.2.2.10.3 ObjectType Description

6.2.2.10.3.1 SafeConditionEquals

SafeConditionEquals indicates which value of the reference variable indicates a save state. Values are

- “TRUE”: indicates that the value “TRUE” of the variable within the PLCOpen XML description represents the safe state of the signal group.
- “FALSE”: indicates that the value “FALSE” of the variable within the PLCOpen XML description represents the safe state of the signal group.

Note: Use of the attribute is optional, default value is TRUE

6.2.2.11 InterlockingConnector

6.2.2.11.1 General

The interface class “InterlockingConnector” shall be used in order to model relations between signal groups and component groups.

6.2.2.11.2 ObjectType Definition

The *InterlockingConnector* is formally defined in Table 38.

Attribute	Value				
BrowseName	InterlockingConnector				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the AutomationMLBaseInterface					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 38: InterlockingConnector Definition

6.2.2.11.3 ObjectType Description

6.2.2.12 Order

6.2.2.12.1 General

The interface class “Order” is an abstract class that shall be used for the description of orders, e.g. a successor or a predecessor.

6.2.2.12.2 ObjectType Definition

The *Order* is formally defined in Table 39.

Attribute	Value				
BrowseName	Order				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the AutomationMLBaseInterface					
HasComponent	Variable	Direction	String	BaseDataVariableType	Optional
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 39: Order Definition

6.2.2.12.3 ObjectType Description

6.2.2.12.3.1 Direction

The attribute “Direction” shall be used in order to specify the direction. Permitted values are “In”, “Out” or “InOut”.

6.2.2.13 PortConnector

6.2.2.13.1 General

The interface class “PortConnector” shall be used in order to provide a high level relation between ports.

6.2.2.13.2 ObjectType Definition

The *PortConnector* is formally defined in Table 40.

Attribute	Value				
BrowseName	PortConnector				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the AutomationMLBaseInterface					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 40: PortConnector Definition

6.2.2.14 PPRConnector

6.2.2.14.1 General

The interface class “PPRConnector” shall be used in order to provide a relation between resources, products and processes.

6.2.2.14.2 ObjectType Definition

The *PPRConnector* is formally defined in Table 41.

Attribute	Value				
BrowseName	PPRConnector				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the AutomationMLBaseInterface					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 41: PPRConnector Definition

6.2.2.14.2.1.1 ObjectTypes

6.3 AutomationMLBaseRoleClassLib

6.3.1 Instances

6.3.1.1 AutomationMLBaseRoleClassLib

This *Instance* is a child of *RoleClassLibs*. It is defined in Table 42.

It consists of the Standard AutomationML Base Role Class Library (Part 1). The content is extended with Part 3 and Part 4 of the AutomationML standard specification.

Name	TypeDefinition
AutomationMLBaseRoleClassLib	FolderType

Table 42: AutomationMLBaseRoleClassLib Instance Definition

6.3.2 ObjectTypes

6.3.2.1 AutomationMLBaseRoleClassLib

6.3.2.1.1 General

The role class “AutomationMLBaseRole” is a basic abstract role type and the base class for all standard or user-defined role classes.

6.3.2.1.2 ObjectType Definition

The *AutomationMLBaseRole* is formally defined in Table 43.

Attribute	Value				
BrowseName	AutomationMLBaseRole				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the CAEXObjectType					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 43: AutomationMLBaseRole Definition

6.3.2.2 Group

6.3.2.2.1 General

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.

6.3.2.2.2 ObjectType Definition

The *Group* is formally defined in Table 44.

Attribute	Value				
BrowseName	Group				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the AutomationMLBaseRole					
HasComponent	Variable	AssociatedFacet	String	BaseDataVariableType	Optional
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 44: Group Definition

6.3.2.2.3 ObjectType Description

6.3.2.2.3.1 AssociatedFacet

The attribute “AssociatedFacet” shall be used for the definition of the name of the corresponding Facet. Example: AssociatedFacet = “PLCFacet”

6.3.2.3 Facet

6.3.2.3.1 General

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.

6.3.2.3.2 ObjectType Definition

The *Facet* is formally defined in Table 45.

Attribute	Value				
BrowseName	Facet				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the AutomationMLBaseRole					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 45: Facet Definition

6.3.2.4 Port

6.3.2.4.1 General

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.

6.3.2.4.2 ObjectType Definition

The *Port* is formally defined in Table 46.

Attribute	Value				
BrowseName	Port				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the AutomationMLBaseRole					
HasComponent	Variable	Cardinality	String	BaseDataVariableType	Optional
HasComponent	Variable	Category	String	BaseDataVariableType	Optional
HasComponent	Object	ConnectionPoint		PortConnector	Optional
HasComponent	Variable	Direction	String	BaseDataVariableType	Optional
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 46: Port Definition

6.3.2.4.3 ObjectType Description

6.3.2.4.3.1 Cardinality

This attribute is a complex attribute which can describe *MaxOccur* or *MinOccur*.

6.3.2.4.3.2 Category

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”

6.3.2.4.3.3 ConnectionPoint

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.

6.3.2.4.3.4 Direction

This attribute shall be used to describe the direction of the Port. Values shall be one of the following: “In” or “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.

6.3.2.5 Resource

6.3.2.5.1 General

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.

6.3.2.5.2 ObjectType Definition

The *Resource* is formally defined in Table 47.

Attribute	Value				
BrowseName	Resource				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the AutomationMLBaseRole					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 47: Resource Definition

6.3.2.6 Product

6.3.2.6.1 General

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.

6.3.2.6.2 ObjectType Definition

The *Product* is formally defined in Table 48.

Attribute	Value				
BrowseName	Product				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the AutomationMLBaseRole					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 48: Product Definition

6.3.2.7 Process

6.3.2.7.1 General

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.

6.3.2.7.2 ObjectType Definition

The *Process* is formally defined in Table 49.

Attribute	Value				
BrowseName	Process				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the AutomationMLBaseRole					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 49: Process Definition

6.3.2.8 Structure

6.3.2.8.1 General

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.

6.3.2.8.2 ObjectType Definition

The *Structure* is formally defined in Table 50.

Attribute	Value				
BrowseName	Structure				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the AutomationMLBaseRole					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 50: Structure Definition

6.3.2.9 ResourceStructure

6.3.2.9.1 General

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

6.3.2.9.2 ObjectType Definition

The *ResourceStructure* is formally defined in Table 51.

Attribute	Value				
BrowseName	ResourceStructure				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Structure					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 51: ResourceStructure Definition

6.3.2.10 ProductStructure

6.3.2.10.1 General

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.

6.3.2.10.2 ObjectType Definition

The *ProductStructure* is formally defined in Table 52.

Attribute	Value				
BrowseName	ProductStructure				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Structure					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 52: *ProductStructure* Definition

6.3.2.11 ProcessStructure**6.3.2.11.1 General**

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.

6.3.2.11.2 ObjectType Definition

The *ProcessStructure* is formally defined in Table 53.

Attribute	Value				
BrowseName	ProcessStructure				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Structure					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 53: *ProcessStructure* Definition

6.3.2.12 Frame**6.3.2.12.1 General**

The role class “Frame” denotes a distinct coordinate system. This coordinate system is explicitly accentuated for special usage.

6.3.2.12.2 ObjectType Definition

The *Frame* is formally defined in Table 54.

Attribute	Value				
BrowseName	Frame				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the AutomationMLBaseRole					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 54: *Frame* Definition

6.3.2.13 ComponentGroup**6.3.2.13.1 General**

The role class “ComponentGroup” is a role type for objects that group objects belonging to one component group according to the interlocking definition of AutomationML.

6.3.2.13.2 ObjectType Definition

The *ComponentGroup* is formally defined in Table 55.

Attribute	Value				
BrowseName	ComponentGroup				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Group					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 55: *ComponentGroup* Definition

6.3.2.14 SignalGroup**6.3.2.14.1 General**

The role class “SignalGroup” is a role type for objects that group objects belonging to one signal group according to the interlocking definition of AutomationML.

6.3.2.14.2 ObjectType Definition

The *SignalGroup* is formally defined in Table 56.

Attribute	Value				
BrowseName	SignalGroup				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Group					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 56: *SignalGroup* Definition

6.3.2.15 PropertySet**6.3.2.15.1 General**

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.

6.3.2.15.2 ObjectType Definition

The *PropertySet* is formally defined in Table 57.

Attribute	Value				
BrowseName	PropertySet				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the AutomationMLBaseRole					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 57: PropertySet Definition

6.4 AutomationMLDMIRoleClassLib

6.4.1 Instances

6.4.1.1 AutomationMLDMIRoleClassLib

This *Instance* is a child of *RoleClassLibs*. It is defined in Table 58.

It consists of the AutomationML Discrete Manufacturing Industry Role Class Library.

Name	TypeDefinition
AutomationMLDMIRoleClassLib	FolderType

Table 58: AutomationMLDMIRoleClassLib Instance Definition

6.4.2 ObjectTypes

6.4.2.1 DiscManufacturingEquipment

6.4.2.1.1 General

The role class “DiscManufacturingEquipment” shall be used for equipment related to discrete manufacturing industries.

6.4.2.1.2 ObjectType Definition

The *DiscManufacturingEquipment* is formally defined in Table 59.

Attribute	Value				
BrowseName	DiscManufacturingEquipment				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Resource					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 59: DiscManufacturingEquipment Definition

6.4.2.2 Transport

6.4.2.2.1 General

The role class “Transport” shall be used for equipment that performs transport processes to transfer items.

6.4.2.2.2 ObjectType Definition

The *Transport* is formally defined in Table 60.

Attribute	Value				
BrowseName	Transport				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the DiscManufacturingEquipment					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 60: Transport Definition

6.4.2.3 Storage

6.4.2.3.1 General

The role class “Storage” shall be used for equipment that is used to buffer products or material temporarily within the plant. It can also be used to feed products or materials into the production process or to export products or materials out of the production process.

6.4.2.3.2 ObjectType Definition

The *Storage* is formally defined in Table 61.

Attribute	Value				
BrowseName	Storage				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the DiscManufacturingEquipment					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 61: Storage Definition

6.4.2.4 Fixture

6.4.2.4.1 General

The role class “Fixture” shall be used for equipment that reduces the degrees of freedom of an item.

6.4.2.4.2 ObjectType Definition

The *Fixture* is formally defined in Table 62.

Attribute	Value				
BrowseName	Fixture				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the DiscManufacturingEquipment					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 62: Fixture Definition

6.4.2.5 Gate

6.4.2.5.1 General

The role class “Gate” shall be used for equipment that can block or monitor an entrance, departure, or a passage way.

6.4.2.5.2 ObjectType Definition

The *Gate* is formally defined in Table 63.

Attribute	Value				
BrowseName	Gate				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the DiscManufacturingEquipment					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 63: Gate Definition

6.4.2.6 Robot

6.4.2.6.1 General

The role class “Robot” shall be used for robots.

6.4.2.6.2 ObjectType Definition

The *Robot* is formally defined in Table 64.

Attribute	Value				
BrowseName	Robot				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the DiscManufacturingEquipment					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 64: Robot Definition

6.4.2.7 Tool

6.4.2.7.1 General

The role class “Tool” shall be used for equipment used by resources that is necessary to or aids in the performance of an operation on the product.

6.4.2.7.2 ObjectType Definition

The *Tool* is formally defined in Table 65.

Attribute	Value				
BrowseName	Tool				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the DiscManufacturingEquipment					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 65: Tool Definition

6.4.2.8 Carrier

6.4.2.8.1 General

The role class “Carrier” shall be used for transport equipment that carries items.

6.4.2.8.2 ObjectType Definition

The *Carrier* is formally defined in Table 66.

Attribute	Value				
BrowseName	Carrier				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the DiscManufacturingEquipment					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 66: Carrier Definition

6.4.2.9 Machine

6.4.2.9.1 General

The role class “Machine” shall be used for mechanic or mechatronic equipment that creates added value on products and is designed expressly to perform specific tasks.

6.4.2.9.2 ObjectType Definition

The *Machine* is formally defined in Table 67.

Attribute	Value				
BrowseName	Machine				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the DiscManufacturingEquipment					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 67: Machine Definition

6.4.2.10 StaticObject

6.4.2.10.1 General

The role class “StaticObject” shall be used for passive, static items positioned in the production environment.

6.4.2.10.2 ObjectType Definition

The *StaticObject* is formally defined in Table 68.

Attribute	Value				
BrowseName	StaticObject				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the DiscManufacturingEquipment					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 68: StaticObject Definition

6.5 AutomationMLCMIRoleClassLib

6.5.1 Instances

6.5.1.1.1 AutomationMLCMIRoleClassLib

This *Instance* is a child of *RoleClassLibs*. It is defined in Table 69.

It consists of the AutomationML Continuous Manufacturing Industry Role Class Library.

Name	TypeDefinition
AutomationMLCMIRoleClassLib	FolderType

Table 69: AutomationMLCMIRoleClassLib Instance Definition

6.5.2 ObjectTypes

6.5.2.1 ContManufacturingEquipment

6.5.2.1.1 General

The role class “ContManufacturingEquipment” shall be used for equipment related to continuous manufacturing industries.

6.5.2.1.2 ObjectType Definition

The *ContManufacturingEquipment* is formally defined in Table 70.

Attribute	Value				
BrowseName	ContManufacturingEquipment				
IsAbstract	False				
References	NodeClass	BrowseName	Data Type	TypeDefinition	ModellingRule
Inherit the components of the CAEXObjectType					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 70: ContManufacturingEquipment Definition

6.6 AutomationMLBMIRoleClassLib

6.6.1 Instances

6.6.1.1 AutomationMLBMIRoleClassLib

This *Instance* is a child of *RoleClassLibs*. It is defined in Table 71.

It consists of the AutomationML Batch Manufacturing Industry Role Class Library.

Name	TypeDefinition
AutomationMLBMIRoleClassLib	FolderType

Table 71: AutomationMLBMIRoleClassLib Instance Definition

6.6.2 ObjectTypes

6.6.2.1 BatchManufacturingEquipment

6.6.2.1.1 General

The role class “BatchManufacturingEquipment” shall be used for equipment related to batch manufacturing industries.

6.6.2.1.2 ObjectType Definition

The *BatchManufacturingEquipment* is formally defined in Table 72.

Attribute	Value				
BrowseName	BatchManufacturingEquipment				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the CAEXObjectType					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 72: BatchManufacturingEquipment Definition

6.7 AutomationMLCSRoleClassLib

6.7.1 Instances

6.7.1.1 AutomationMLCSRoleClassLib

This *Instance* is a child of *RoleClassLibs*. It is defined in Table 73.

It consists of the AutomationML Control Industry Role Class Library.

Name	TypeDefinition
AutomationMLCSRoleClassLib	FolderType

Table 73: AutomationMLCSRoleClassLib Instance Definition

6.7.2 ObjectTypes

6.7.2.1 ControlEquipment

6.7.2.1.1 General

The role class “ControlEquipment” shall be used for equipment related to a control system. ControlEquipment can be used for every type of industry.

6.7.2.1.2 ObjectType Definition

The *ControlEquipment* is formally defined in Table 74.

Attribute	Value				
BrowseName	ControlEquipment				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Resource					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 74: ControlEquipment Definition

6.7.2.2 Communication

6.7.2.2.1 General

6.7.2.2.2 ObjectType Definition

The *Communication* is formally defined in Table 75.

Attribute	Value				
BrowseName	Communication				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the ControlEquipment					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 75: Communication Definition

6.7.2.3 ControlHardware

6.7.2.3.1 General

The role class “ControlHardware” shall be used for hardware that provides runtime environments.

6.7.2.3.2 ObjectType Definition

The *ControlHardware* is formally defined in Table 76.

Attribute	Value				
BrowseName	ControlHardware				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the ControlEquipment					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 76: ControlHardware Definition

6.7.2.4 PC

6.7.2.4.1 General

The role class “PC” shall be used for any general-purpose computer that provides runtime environments for software being executed on it.

6.7.2.4.2 ObjectType Definition

The *PC* is formally defined in Table 77.

Attribute	Value				
BrowseName	PC				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the ControlHardware					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 77: PC Definition

6.7.2.5 IPC

6.7.2.5.1 General

The role class “IPC” shall be used for any PC-based computing platform for industrial applications that provides runtime environments for software being executed on it.

6.7.2.5.2 ObjectType Definition

The *IPC* is formally defined in Table 78.

Attribute	Value				
BrowseName	IPC				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the ControlHardware					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 78: IPC Definition

6.7.2.6 Handheld

6.7.2.6.1 General

The role class “Handheld” shall be used for any portable, programmable, electronic device with an own power supply for particular applications.

6.7.2.6.2 ObjectType Definition

The *Handheld* is formally defined in Table 79.

Attribute	Value				
BrowseName	Handheld				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the ControlHardware					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 79: Handheld Definition

6.7.2.7 EmbeddedDevice

6.7.2.7.1 General

The role class “EmbeddedDevice” shall be used for any device designed to perform one or a few dedicated software functions. It is embedded as part of a complete device often including hardware and mechanical parts.

6.7.2.7.2 ObjectType Definition

The *EmbeddedDevice* is formally defined in Table 80.

Attribute	Value				
BrowseName	EmbeddedDevice				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the ControlHardware					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 80: EmbeddedDevice Definition

6.7.2.8 Sensor

6.7.2.8.1 General

The role class “Sensor” shall be used for sensors.

6.7.2.8.2 ObjectType Definition

The *Sensor* is formally defined in Table 81.

Attribute	Value				
BrowseName	Sensor				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the ControlEquipment					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 81: Sensor Definition

6.7.2.9 Actuator**6.7.2.9.1 General**

The role class “Actuator” shall be used for actuators.

6.7.2.9.2 ObjectType Definition

The *Actuator* is formally defined in Table 82.

Attribute	Value				
BrowseName	Actuator				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the ControlEquipment					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 82: Actuator Definition

6.7.2.10 Controller**6.7.2.10.1 General**

The role class “Controller” shall be used for self-acting functionalities that process signals according to a predefined logic and generate output signals in order to reach an intended behaviour of technical processes.

6.7.2.10.2 ObjectType Definition

The *Controller* is formally defined in Table 83.

Attribute	Value				
BrowseName	Controller				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the ControlEquipment					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 83: Controller Definition

6.7.2.11 PLC**6.7.2.11.1 General**

The role class “PLC” shall be used for programmable control functionality focusing the processing of signals.

6.7.2.11.2 ObjectType Definition

The *PLC* is formally defined in Table 84.

Attribute	Value				
BrowseName	PLC				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Controller					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 84: PLC Definition

6.7.2.12 NC**6.7.2.12.1 General**

The role class “NC” shall be used for programmable control functionality focusing the processing of numerical signals.

6.7.2.12.2 ObjectType Definition

The *NC* is formally defined in Table 85.

Attribute	Value				
BrowseName	NC				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Controller					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 85: NC Definition

6.7.2.13 RC**6.7.2.13.1 General**

The role class “RC” shall be used for programmable control functionality driving robots in order to reach an intended behaviour of the robot kinematic system and corresponding connected periphery.

6.7.2.13.2 ObjectType Definition

The *RC* is formally defined in Table 86.

Attribute	Value				
BrowseName	RC				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Controller					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 86: RC Definition

6.7.2.14 PAC

6.7.2.14.1 General

The role class “PAC” shall be used for programmable automation functionality focusing on cross-domain functionality like binary, motion and continuous control.

6.7.2.14.2 ObjectType Definition

The *PAC* is formally defined in Table 87.

Attribute	Value				
BrowseName	PAC				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Controller					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 87: PAC Definition

6.8 AutomationMLExtendedRoleClassLib

6.8.1 Instances

6.8.1.1 AutomationMLExtendedRoleClassLib

This *Instance* is a child of *RoleClassLibs*. It is defined in Table 88.

The *AutomationMLExtendedRoleClassLibrary* is a recommended extension of the *AutomationMLBaseRoleClassLib* and the *AutomationMLDMIRoleClassLib* and covers a wide area of typical roles of the discrete manufacturing industry.

Name	TypeDefinition
AutomationMLExtendedRoleClassLib	FolderType

Table 88: AutomationMLExtendedRoleClassLib Instance Definition

6.8.2 ObjectTypes

6.8.2.1 PLCFacet

6.8.2.1.1 General

The role class “PLCFacet” should be used for the view concerning everything involved in PLC control code generators: PLC view on AML objects which points to information concerning PLC.

6.8.2.1.2 ObjectType Definition

The *PLCFacet* is formally defined in Table 89.

Attribute	Value				
BrowseName	PLCFacet				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Facet					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 89: PLCFacet Definition

6.8.2.2 HMIFacet

6.8.2.2.1 General

The role class “HMIFacet” should be used for the view concerning everything involved in HMI: HMI view on AML objects which points to information concerning HMI.

6.8.2.2.2 ObjectType Definition

The *HMIFacet* is formally defined in Table 90.

Attribute	Value				
BrowseName	HMIFacet				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Facet					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 90: HMIFacet Definition

6.8.2.3 Enterprise

6.8.2.3.1 General

The role class “Enterprise” should be used for business structures. The definition of an “Enterprise” is given in IEC 62264-1:2013, 5.3.2: “An enterprise is a collection of sites and areas and represents the top level of a role based equipment hierarchy. The enterprise is responsible for determining what products will be manufactured, at which sites they will be manufactured, and in general how they will be manufactured. Level 4 functions are generally concerned with the enterprise and site levels. However, enterprise planning and scheduling may involve areas, work centers, or work units within an area.”

6.8.2.3.2 ObjectType Definition

The *Enterprise* is formally defined in Table 91.

Attribute	Value				
BrowseName	Enterprise				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the ResourceStructure					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 91: Enterprise Definition

6.8.2.4 Site

6.8.2.4.1 General

The role class “Site” should be used for localisation. It is also used for hierarchical organization. The definition of a “Site” is given in IEC 62264-1:2013, 5.3.3: “A site is a physical, geographical, or logical grouping determined by the enterprise. It may contain areas, production lines, process cells, and production units. Site planning and scheduling may involve cells, lines, or units within the areas. A geographical location and main production capability usually identifies a site. Examples of site identifications are ‘Deer Park Olefins Plant’ and ‘Johnson City Manufacturing Facility’. Sites are often used for rough-cut planning and scheduling. Sites generally have well-defined manufacturing capabilities.

6.8.2.4.2 ObjectType Definition

The *Site* is formally defined in Table 92.

Attribute	Value				
BrowseName	Site				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the ResourceStructure					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 92: Site Definition

6.8.2.5 Area

6.8.2.5.1 General

The role class “Area” should be used for production buildings and their subdivisions (structure/hall), used for hierarchical organization. The definition of an “Area” is given in IEC 62264-1:2013, 5.3.4: “An area is a physical, geographical, or logical grouping determined by the site. It may contain process cells, production units, and production lines. The main production capability and geographical location within a site usually identify areas. Examples of area identifications are ‘North End Tank Farm’ and ‘Building 2 Electronic Assembly’. Areas generally have well-defined manufacturing capabilities and capacities. The capabilities and capacities are used for planning and scheduling. An area is made up of lower-level elements that perform the manufacturing functions. There are three types of elements defined that correspond to continuous manufacturing models, discrete (repetitive and nonrepetitive) manufacturing models, and batch manufacturing models. An area may have one or more of any of the lower-level elements depending upon the manufacturing requirements. Many areas will have a combination of production lines for the discrete operations, production units for the continuous processes, and process cells for batch processes. For example, a beverage manufacturer may have an area with continuous mixing in a production unit, which feeds a batch process cell for batch processing, feeding a bottling line for a discrete bottling process.”

6.8.2.5.2 ObjectType Definition

The *Area* is formally defined in Table 93.

Attribute	Value				
BrowseName	Area				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the ResourceStructure					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 93: Area Definition

6.8.2.6 ProductionLine

6.8.2.6.1 General

The role class “ProductionLine” should be used for defining the role based equipment hierarchy defined in IEC 62264-1:2013, 5.3.7, for discrete manufacturing at the work cell level: “Production lines and work cells are the lowest level of equipment. Work cells are usually only identified when there is flexibility in the routing of work within a production line. Production lines and work cells may be composed of lower-level elements, but definitions of these are outside the scope of this document. The major processing activity often identifies the production line. Examples of production line identifications are ‘Bottling Line #1’, ‘Capping Line #15’, and ‘Water Pump Assembly Line #4’. Production line and work cells have well-defined manufacturing capabilities and throughput capacities.”

6.8.2.6.2 ObjectType Definition

The *ProductionLine* is formally defined in Table 94.

Attribute	Value				
BrowseName	ProductionLine				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the ResourceStructure					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 94: ProductionLine Definition

6.8.2.7 WorkCell

6.8.2.7.1 General

The role class “WorkCell” should be used for defining the role based equipment hierarchy defined in IEC 62264-1:2013 at the work cell level: for sub units/sub production steps of units/production lines, station, processes single component, cycle, location in which the production step takes place. It is used for hierarchization. The definition of a “WorkCell” is given in IEC 62264-1:2013, 5.3.7: “Production lines and work cells are the lowest level of equipment. Work cells are usually only identified when there is flexibility in the routing of work within a production line. Production lines and work cells may be composed of lower-level elements, but definitions of these are outside the scope of this document. The major processing activity often identifies the production line. Examples of production line identifications are ‘Bottling Line #1’, ‘Capping Line #15’, and ‘Water Pump Assembly Line #4’. Production line and work cells have well-defined manufacturing capabilities and throughput capacities.”

6.8.2.7.2 ObjectType Definition

The *WorkCell* is formally defined in Table 95.

Attribute	Value				
BrowseName	WorkCell				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the ResourceStructure					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 95: WorkCell Definition

6.8.2.8 ProcessCell

6.8.2.8.1 General

The role class “ProcessCell” should be used for sub units/sub production steps of units/production lines, station, processes single component, cycle, location in which the production step takes place. It is used for hierarchization. The definition of a “ProcessCell” is given in IEC 62264-1:2013, 5.3.8: “Process cells and units are the lowest level of equipment for batch manufacturing processes. Units are usually only identified if there is flexibility in the routing of product within a process cell. The definitions for process cells and units are contained in the IEC 61512-1 standard. The major processing capability or family of products produced often identifies the process cell. Examples of process cell identifications are ‘Mixing Line #5’ and ‘Detergent Line 13’. Process cells and units have well-defined manufacturing capabilities and batch capacities.”

6.8.2.8.2 ObjectType Definition

The *ProcessCell* is formally defined in Table 96.

Attribute	Value				
BrowseName	ProcessCell				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the ResourceStructure					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 96: ProcessCell Definition

6.8.2.9 Unit

6.8.2.9.1 General

The role class “Unit” should be used for linked chained production plants. It is used for hierarchization. The definition of a “Unit” is given in IEC 62264-1:2013, 5.3.8: “Process cells and units are the lowest level of equipment for batch manufacturing processes. Units are usually only identified if there is flexibility in the routing of product within a process cell. The definitions for process cells and units are contained in the IEC 61512-1 standard. The major processing capability or family of products produced often identifies the process cell. Examples of process cell identifications are ‘Mixing Line #5’ and ‘Detergent Line 13’. Process cells and units have well-defined manufacturing capabilities and batch capacities.”

6.8.2.9.2 ObjectType Definition

The *Unit* is formally defined in Table 97.

Attribute	Value				
BrowseName	Unit				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the ResourceStructure					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 97: Unit Definition

6.8.2.10 ProductionUnit

6.8.2.10.1 General

The role class “ProductionUnit” should be used for sub units/sub production steps of units/production lines, station, processes single component, cycle, location in which the production step takes place. It is used for hierarchization. The definition of a “ProductionUnit” is given in IEC 62264-1:2013, 5.3.6: “Production units are the lowest level of equipment for continuous manufacturing processes. Production units are composed of lower level elements, such as equipment modules, sensors, and actuators, but definitions of these are outside the scope of the IEC 62714 series. A production unit generally encompasses all of the equipment required for a segment of continuous production that operates in a relatively autonomous manner. It generally converts, separates, or reacts to one or more feedstocks to produce intermediate or final products. The major processing activity or product generated often identifies the production unit. Examples of production unit identifications are ‘Catalytic Cracker #1’ and ‘Alkylation Unit 2’. Production units have well-defined processing capabilities and throughput capacities.”

6.8.2.10.2 ObjectType Definition

The *ProductionUnit* is formally defined in Table 98.

Attribute	Value				
BrowseName	ProductionUnit				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the ResourceStructure					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 98: ProductionUnit Definition

6.8.2.11 StorageZone

6.8.2.11.1 General

The role class “StorageZone” should be used defining the role based equipment hierarchy defined in IEC 62264-1:2013 at the storage zone level: The definition of an “StorageZone” is given in IEC 62264-1: 2013, 5.3.9: “Storage zones and storage units are the lowest level of material movement equipment typically scheduled by the Level 4 and Level 3 functions for discrete, batch and continuous manufacturing processes. A storage zone is a type of work center and a storage unit is a type of work unit that is organized as elements within an area. These are the lower-level elements of an equipment hierarchy used in material storage and movement activities. A storage zone typically has the capability needed for the receipt, storage, retrieval, movement and shipment of materials. This may include the movement of materials from one work center to another work center within or between enterprises.”

6.8.2.11.2 ObjectType Definition

The *StorageZone* is formally defined in Table 99.

Attribute	Value				
BrowseName	StorageZone				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the ResourceStructure					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 99: StorageZone Definition

6.8.2.12 StorageUnit

6.8.2.12.1 General

The role class “StorageUnit” should be used defining the role based equipment hierarchy defined in IEC 62264-1:2013 at the storage unit level: The definition of an “StorageUnit” is given in IEC 62264-1: 2013, 5.3.9: “Storage zones and storage units are the lowest level of material movement equipment typically scheduled by the Level 4 and Level 3 functions for discrete, batch and continuous manufacturing processes. Storage units are typically managed at a finer level of detail than a storage zone. The physical location of a storage unit may change over time; for example, for goods in transit. Storage units may be dedicated to a given material, group of materials, or method of storage. Storage units can be further divided to address any hierarchical storage management scheme.”

6.8.2.12.2 ObjectType Definition

The *StorageUnit* is formally defined in Table 100.

Attribute	Value				
BrowseName	StorageUnit				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the ResourceStructure					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 100: StorageUnit Definition

6.8.2.13 Turntable

6.8.2.13.1 General

The role class “Turntable” should be used for rotating transport equipment which changes the horizontal transport direction of a product and/or carrier.

6.8.2.13.2 ObjectType Definition

The *Turntable* is formally defined in Table 101.

Attribute	Value				
BrowseName	Turntable				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Transport					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 101: Turntable Definition

6.8.2.14 Conveyor

6.8.2.14.1 General

The role class “Conveyor” should be used for generic equipment which performs linear transport.

6.8.2.14.2 ObjectType Definition

The *Conveyor* is formally defined in Table 102.

Attribute	Value				
BrowseName	Conveyor				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Transport					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 102: Conveyor Definition

6.8.2.15 BeltConveyor

6.8.2.15.1 General

The role class “BeltConveyor” should be used for equipment which performs linear transport realized by one or more belts as transport platform.

6.8.2.15.2 ObjectType Definition

The *BeltConveyor* is formally defined in Table 103.

Attribute	Value				
BrowseName	BeltConveyor				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Conveyor					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 103: BeltConveyor Definition

6.8.2.16 RollConveyor

6.8.2.16.1 General

The role class “RollConveyor” should be used for equipment which performs linear transport realized by a sequence of rolls as transport platform.

6.8.2.16.2 ObjectType Definition

The *RollConveyor* is formally defined in Table 104.

Attribute	Value				
BrowseName	RollConveyor				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Conveyor					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 104: RollConveyor Definition

6.8.2.17 ChainConveyor**6.8.2.17.1 General**

The role class “ChainConveyor” should be used for equipment which performs linear transport driven by an endless chain as transport medium.

6.8.2.17.2 ObjectType Definition

The *ChainConveyor* is formally defined in Table 105.

Attribute	Value				
BrowseName	ChainConveyor				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Conveyor					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 105: ChainConveyor Definition

6.8.2.18 PalletConveyor**6.8.2.18.1 General**

The role class “PalletConveyor” should be used for equipment which is especially designed for linear transport of pallets.

6.8.2.18.2 ObjectType Definition

The *PalletConveyor* is formally defined in Table 106.

Attribute	Value				
BrowseName	PalletConveyor				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Conveyor					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 106: PalletConveyor Definition

6.8.2.19 OverheadConveyor**6.8.2.19.1 General**

The role class “OverheadConveyor” should be used for equipment that performs overhead transport of hanging products or carriers.

6.8.2.19.2 ObjectType Definition

The *OverheadConveyor* is formally defined in Table 107.

Attribute	Value				
BrowseName	OverheadConveyor				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Conveyor					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 107: *OverheadConveyor* Definition

6.8.2.20 LiftingTable**6.8.2.20.1 General**

The role class “LiftingTable” should be used for equipment that performs discrete vertical transport. The transport medium is also lifted. Normally used for minor heights.

6.8.2.20.2 ObjectType Definition

The *LiftingTable* is formally defined in Table 108.

Attribute	Value				
BrowseName	LiftingTable				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Transport					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 108: *LiftingTable* Definition

6.8.2.21 AGV**6.8.2.21.1 General**

The role class “AGV” should be used for equipment that performs automated transportation of discrete units independent of other transport equipment.

6.8.2.21.2 ObjectType Definition

The AGV is formally defined in Table 109.

Attribute	Value				
BrowseName	AGV				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Transport					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 109: AGV Definition

6.8.2.22 Transposer

6.8.2.22.1 General

The role class “Transposer” should be used for transport equipment that performs the change of the transport medium. Changes the classification or relation of product to the carrier (one to another).

6.8.2.22.2 ObjectType Definition

The *Transposer* is formally defined in Table 110.

Attribute	Value				
BrowseName	Transposer				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Transport					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 110: Transposer Definition

6.8.2.23 CarrierHandlingSystem

6.8.2.23.1 General

The role class “CarrierHandlingSystem” should be used for equipment that performs an action to the carrier.

6.8.2.23.2 ObjectType Definition

The *CarrierHandlingSystem* is formally defined in Table 111.

Attribute	Value				
BrowseName	CarrierHandlingSystem				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Transport					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 111: CarrierHandlingSystem Definition

6.8.2.24 BodyStore

6.8.2.24.1 General

The role class “BodyStore” should be used for buffering discrete products.

6.8.2.24.2 ObjectType Definition

The *BodyStore* is formally defined in Table 112.

Attribute	Value				
BrowseName	BodyStore				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Storage					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 112: *BodyStore* Definition

6.8.2.25 Lift**6.8.2.25.1 General**

The role class “Lift” should be used for equipment that performs discrete vertical transport. Normally used for larger heights.

6.8.2.25.2 ObjectType Definition

The *Lift* is formally defined in Table 113.

Attribute	Value				
BrowseName	Lift				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Transport					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 113: *Lift* Definition

6.8.2.26 Rollerbed**6.8.2.26.1 General**

The role class “Rollerbed” should be used for a sequence of rolls. None of these rolls are driven.

6.8.2.26.2 ObjectType Definition

The *Rollerbed* is formally defined in Table 114.

Attribute	Value				
BrowseName	Rollerbed				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Transport					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 114: *Rollerbed* Definition

6.8.2.27 StationaryTool**6.8.2.27.1 General**

The role class “StationaryTool” should be used for tools fixed at one place.

6.8.2.27.2 ObjectType Definition

The *StationaryTool* is formally defined in Table 115.

Attribute	Value				
BrowseName	StationaryTool				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Tool					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 115: *StationaryTool* Definition

6.8.2.28 MovableTool**6.8.2.28.1 General**

The role class “MovableTool” should be used for tools which can be moved by equipment e.g. robots.

6.8.2.28.2 ObjectType Definition

The *MovableTool* is formally defined in Table 116.

Attribute	Value				
BrowseName	MovableTool				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Tool					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 116: *MovableTool* Definition

6.8.2.29 ControlCabinet**6.8.2.29.1 General**

The role class “ControlCabinet” should be used for enclosed electrical and/or electronic assembly.

6.8.2.29.2 ObjectType Definition

The *ControlCabinet* is formally defined in Table 117.

Attribute	Value				
BrowseName	ControlCabinet				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the ControlEquipment					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 117: *ControlCabinet* Definition

6.8.2.30 IODevice**6.8.2.30.1 General**

The role class “IODevice” should be used for devices providing the functionality to connect sensors or actuators with an automation system. IODevice can consist of different modules.

6.8.2.30.2 ObjectType Definition

The *IODevice* is formally defined in Table 118.

Attribute	Value				
BrowseName	IODevice				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the ControlEquipment					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 118: *IODevice* Definition

6.8.2.31 HMI**6.8.2.31.1 General**

The role class “HMI” should be used for the functionality to visualize an industrial control and monitoring system for the effective operation and control of the machine by humans.

6.8.2.31.2 ObjectType Definition

The *HMI* is formally defined in Table 119.

Attribute	Value				
BrowseName	HMI				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the ControlEquipment					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 119: *HMI* Definition

6.8.2.32 WarningEquipment**6.8.2.32.1 General**

The role class “WarningEquipment” should be used for equipment providing warning functionality. NOTE The functionality can be realized in auditive, visual, haptic or other way.

6.8.2.32.2 ObjectType Definition

The *WarningEquipment* is formally defined in Table 120.

Attribute	Value				
BrowseName	WarningEquipment				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the HMI					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 120: *WarningEquipment* Definition

6.8.2.33 ActuatingDrive**6.8.2.33.1 General**

The role class “ActuatingDrive” should be used for physical unit used for driving mechanically actuated final controlling elements.

6.8.2.33.2 ObjectType Definition

The *ActuatingDrive* is formally defined in Table 121.

Attribute	Value				
BrowseName	ActuatingDrive				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Actuator					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 121: *ActuatingDrive* Definition

6.8.2.34 MotionController**6.8.2.34.1 General**

The role class “MotionController” should be used for logic to generate set points (the desired output or motion profile) and close a position or velocity feedback loop.

6.8.2.34.2 ObjectType Definition

The *MotionController* is formally defined in Table 122.

Attribute	Value				
BrowseName	MotionController				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the ControlEquipment					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 122: *MotionController* Definition

6.8.2.35 Panel**6.8.2.35.1 General**

The role class “Panel” should be used for physical object providing one possibility for humans to interact with machines.

6.8.2.35.2 ObjectType Definition

The *Panel* is formally defined in Table 123.

Attribute	Value				
BrowseName	Panel				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the ControlHardware					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 123: Panel Definition

6.8.2.36 MeasuringEquipment

6.8.2.36.1 General

The role class “MeasuringEquipment” should be used for defining equipment defined in IEC60050-311:2001, 311-03-05: “assembly of measuring instruments intended for specified measurement purposes”

6.8.2.36.2 ObjectType Definition

The *MeasuringEquipment* is formally defined in Table 124.

Attribute	Value				
BrowseName	MeasuringEquipment				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Resource					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 124: MeasuringEquipment Definition

6.8.2.37 Clamp

6.8.2.37.1 General

The role class “Clamp” should be used for equipment that performs fixation processes to hold items at one specific point.

6.8.2.37.2 ObjectType Definition

The *Clamp* is formally defined in Table 125.

Attribute	Value				
BrowseName	Clamp				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Fixture					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 125: Clamp Definition

6.8.2.38 ProcessController

6.8.2.38.1 General

The role class “ProcessController” should be used for the control of a specific tool or machine that performs process steps on a product.

6.8.2.38.2 ObjectType Definition

The *ProcessController* is formally defined in Table 126.

Attribute	Value				
BrowseName	ProcessController				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Controller					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 126: *ProcessController* Definition

6.8.2.39 Loader**6.8.2.39.1 General**

The role class “Loader” should be used for equipment to introduce products into the production process.

6.8.2.39.2 ObjectType Definition

The *Loader* is formally defined in Table 127.

Attribute	Value				
BrowseName	Loader				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Storage					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 127: *Loader* Definition

6.8.2.40 Unloader**6.8.2.40.1 General**

The role class “Unloader” should be used for equipment to export products out of the production process.

6.8.2.40.2 ObjectType Definition

The *Unloader* is formally defined in Table 128.

Attribute	Value				
BrowseName	Unloader				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherit the components of the Storage					
HasProperty	Variable	ID	BaseDataType	PropertyType	Optional
HasProperty	Variable	Version	BaseDataType	PropertyType	Optional

Table 128: Unloader Definition

7 Profiles

7.1 OPC UA Conformance Units and Profiles

This chapter defines the corresponding profiles and conformance units for the OPC UA Information Model for AutomationML. Profiles are named groupings of conformance units. Facets are profiles that will be combined with other *Profiles* to define the complete functionality of an OPC UA *Server* or *Client*.

Table 129 specify the facet available for *Servers* that implement the AutomationML Information Model companion specification.

Conformance Unit	Description	Optional/ Mandatory
AML Information Model	Support Objects that conform to the AutomationML base types specified in the AutomationML companion standard in chapter 5. This includes in particular all types derived from CAEXBasicObjectType.	M
AML Libraries	Server provides additional data for AutomationML Libraries defined in chapter 6.	O

Table 129: AutomationML Server Facet Definition

Table 130 defines the facet available for *Clients* that implement the AutomationML Information Model companion specification.

Conformance Unit	Description	Optional/ Mandatory
AML Client Information Model	Consume Objects that conform to the AutomationML base types specified in the AutomationML companion standard in chapter 5. This includes in particular all types derived from CAEXBasicObjectType.	M
AML Client Libraries	Use available additional data for AutomationML Libraries defined in chapter 6.	O

Table 130: AutomationML Client Facet Definition

8 Address space structure

8.1 Handling of OPC UA namespaces

Namespaces are used by OPC UA to create unique identifiers across different naming authorities. The *Attributes NodeId* and *BrowseName* are identifiers. A node in the UA *Address Space* 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* shall have the namespace of the standards body although 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.

Table 131 provides a list of mandatory and optional namespaces used in an AutomationML OPC UA Server.

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

Table 131: Namespaces used in a AutomationML Server

Annex A (informative): Mapping example

```

<?xml version="1.0" encoding="utf-8"?>
<CAEXFile      FileName="Topology.aml"      SchemaVersion="2.15"
xsi:noNamespaceSchemaLocation="./Source/CAEX_ClassModel_V2.15.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <AdditionalInformation AutomationMLVersion="2.0"/>
  <AdditionalInformation>
    <WriterHeader>
      <WriterName>AutomationML e.V.</WriterName>
      <WriterID>AutomationML e.V.</WriterID>
      <WriterVendor>AutomationML e.V.</WriterVendor>
      <WriterVendorURL>www.AutomationML.org</WriterVendorURL>
      <WriterVersion>1.0</WriterVersion>
      <WriterRelease>1.0</WriterRelease>
      <LastWritingDateTime>2012-02-20
      </LastWritingDateTime>
      <WriterProjectTitle>AutomationML
Examples</WriterProjectTitle>
      <WriterProjectID>AutomationML Tutorial Examples
      </WriterProjectID>
    </WriterHeader>
    </AdditionalInformation>
    <ExternalReference      Path="Libs/RoleClass
Libraries/AutomationMLBaseRoleClassLib.aml" Alias="BaseRoleClassLib"/>
    <ExternalReference      Path="Libs/InterfaceClass
Libraries/AutomationMLInterfaceClassLib.aml" Alias="BaseInterfaceClassLib"/>
    <InstanceHierarchy Name="ManufacturingSystem">
      <InternalElement Name="firstScrewdriver" ID="{788eb291-f103-4fdc-aba0-
4893b599f556}" RefBaseSystemUnitPath="LibOfCommonTools/ElectricScrewdriver">
        <Attribute Name="New Attribute"/>
        <ExternalInterface Name="EnergySupply" ID="{5f535d4c-dd46-4c1c-
898c-4e58419048b6}" RefBaseClassPath="MyInterfaces/Energy"/>
        <SupportedRoleClass
RefRoleClassPath="ManufacturingRoleClasses/Tool"/>
        <RoleRequirements
RefBaseRoleClassPath="ManufacturingRoleClasses/Tool"/>
      </InternalElement>
      <InternalElement Name="secondScrewdriver" ID="{19dcf818-4716-4fc1-a85f-
28e1938c4c3a}" RefBaseSystemUnitPath="LibOfCommonTools/ElectricScrewdriver">
        <ExternalInterface Name="EnergySupply" ID="{50e10905-ac18-413c-
afab-ad8ed1569fff}" RefBaseClassPath="MyInterfaces/Energy"/>
        <SupportedRoleClass
RefRoleClassPath="ManufacturingRoleClasses/Tool"/>
        <RoleRequirements
RefBaseRoleClassPath="ManufacturingRoleClasses/Tool"/>
      </InternalElement>
    </InstanceHierarchy>
    <InterfaceClassLib Name="MyInterfaces">
      <Version>1.0</Version>
      <InterfaceClass      Name="Energy"
RefBaseClassPath="BaseInterfaceClassLib@AutomationMLInterfaceClassLib/AutomationMLB
aseInterface"/>
    </InterfaceClassLib>
    <RoleClassLib Name="ManufacturingRoleClasses">
      <Version>1.0</Version>
      <RoleClass      Name="Tool"
RefBaseClassPath="BaseRoleClassLib@AutomationMLBaseRoleClassLib/AutomationMLBase
Role"/>
    </RoleClassLib>
    <SystemUnitClassLib Name="LibOfCommonTools">

```



```

    <Version>1.0</Version>
    <SystemUnitClass Name="ElectricScrewdriver">
      <ExternalInterface Name="EnergySupply" ID="dd0e0dfe-10f8-4068-
845b-9c29699ac79b" RefBaseClassPath="MyInterfaces/Energy"/>
      <SupportedRoleClass
RefRoleClassPath="ManufacturingRoleClasses/Tool"/>
    </SystemUnitClass>
  </SystemUnitClassLib>
</CAEXFile>

```

Figure 21: AutomationML XML text

```

<?xml version="1.0" encoding="utf-8"?>
<UANodeSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://opcfoundation.org/UA/2011/03/UANodeSet.xsd">
  <NamespaceUris>
    <Uri>http://opcfoundation.org/UA/AML/</Uri>
    <Uri>http://www.iosb.fraunhofer.de/Topology.aml</Uri>
  </NamespaceUris>
  <Aliases>
    <Alias Alias="Boolean">i=1</Alias>
    <Alias Alias="SByte">i=2</Alias>
    <Alias Alias="Byte">i=3</Alias>
    <Alias Alias="Int16">i=4</Alias>
    <Alias Alias="UInt16">i=5</Alias>
    <Alias Alias="Int32">i=6</Alias>
    <Alias Alias="UInt32">i=7</Alias>
    <Alias Alias="Int64">i=8</Alias>
    <Alias Alias="UInt64">i=9</Alias>
    <Alias Alias="Float">i=10</Alias>
    <Alias Alias="Double">i=11</Alias>
    <Alias Alias="DateTime">i=13</Alias>
    <Alias Alias="String">i=12</Alias>
    <Alias Alias="ByteString">i=15</Alias>
    <Alias Alias="Guid">i=14</Alias>
    <Alias Alias="XmlElement">i=16</Alias>
    <Alias Alias="NodeId">i=17</Alias>
    <Alias Alias="ExpandedNodeId">i=18</Alias>
    <Alias Alias="QualifiedName">i=20</Alias>
    <Alias Alias="LocalizedText">i=21</Alias>
    <Alias Alias="StatusCode">i=19</Alias>
    <Alias Alias="Structure">i=22</Alias>
    <Alias Alias="Number">i=26</Alias>
    <Alias Alias="Integer">i=27</Alias>
    <Alias Alias="UInteger">i=28</Alias>
    <Alias Alias="HasComponent">i=47</Alias>
    <Alias Alias="HasProperty">i=46</Alias>
    <Alias Alias="Organizes">i=35</Alias>
    <Alias Alias="HasEventSource">i=36</Alias>
    <Alias Alias="HasNotifier">i=48</Alias>
    <Alias Alias="HasSubtype">i=45</Alias>
    <Alias Alias="HasTypeDefinition">i=40</Alias>
    <Alias Alias="HasModellingRule">i=37</Alias>
    <Alias Alias="HasEncoding">i=38</Alias>
    <Alias Alias="HasDescription">i=39</Alias>
    <Alias Alias="Duration">i=290</Alias>
  </Aliases>
  <UAObject NodeId="ns=2;i=1" BrowseName="ManufacturingSystem">
    <DisplayName>ManufacturingSystem</DisplayName>
    <Description></Description>

```

```

<References>
  <Reference ReferenceType="HasTypeDefinition">i=61</Reference>
  <Reference ReferenceType="Organizes" IsForward="false">ns=1;i=5005</Reference>
  <Reference ReferenceType="HasComponent" IsForward="false">ns=2;i=25</Reference>
  <Reference ReferenceType="HasComponent">ns=2;i=8</Reference>
  <Reference ReferenceType="HasComponent">ns=2;i=10</Reference>
</References>
</UAObject>
<UAObject NodeId="ns=2;i=2" BrowseName="MyInterfaces">
  <DisplayName>MyInterfaces</DisplayName>
  <Description></Description>
  <References>
    <Reference ReferenceType="HasProperty">ns=2;i=3</Reference>
    <Reference ReferenceType="HasTypeDefinition">i=61</Reference>
    <Reference ReferenceType="Organizes" IsForward="false">ns=1;i=5008</Reference>
    <Reference ReferenceType="HasComponent" IsForward="false">ns=2;i=28</Reference>
    <Reference ReferenceType="ns=1;i=4002">ns=2;i=17</Reference>
  </References>
</UAObject>
<UAObject NodeId="ns=2;i=4" BrowseName="ManufacturingRoleClasses">
  <DisplayName>ManufacturingRoleClasses</DisplayName>
  <Description></Description>
  <References>
    <Reference ReferenceType="HasProperty">ns=2;i=5</Reference>
    <Reference ReferenceType="HasTypeDefinition">i=61</Reference>
    <Reference ReferenceType="Organizes" IsForward="false">ns=1;i=5009</Reference>
    <Reference ReferenceType="HasComponent" IsForward="false">ns=2;i=27</Reference>
    <Reference ReferenceType="ns=1;i=4002">ns=2;i=18</Reference>
  </References>
</UAObject>
<UAObject NodeId="ns=2;i=6" BrowseName="LibOfCommonTools">
  <DisplayName>LibOfCommonTools</DisplayName>
  <Description></Description>
  <References>
    <Reference ReferenceType="HasProperty">ns=2;i=7</Reference>
    <Reference ReferenceType="HasTypeDefinition">i=61</Reference>
    <Reference ReferenceType="Organizes" IsForward="false">ns=1;i=5010</Reference>
    <Reference ReferenceType="HasComponent" IsForward="false">ns=2;i=26</Reference>
    <Reference ReferenceType="ns=1;i=4002">ns=2;i=19</Reference>
  </References>
</UAObject>
<UAObject NodeId="ns=2;i=8" BrowseName="firstScrewdriver">
  <DisplayName>firstScrewdriver</DisplayName>
  <Description></Description>
  <References>
    <Reference ReferenceType="HasProperty">ns=2;i=9</Reference>
    <Reference ReferenceType="HasComponent" IsForward="false">ns=2;i=1</Reference>
    <Reference ReferenceType="HasTypeDefinition">ns=2;i=19</Reference>
    <Reference ReferenceType="ns=1;i=4001">ns=2;i=18</Reference>
    <Reference ReferenceType="ns=1;i=4001">ns=2;i=18</Reference>
    <Reference ReferenceType="HasComponent">ns=2;i=12</Reference>
    <Reference ReferenceType="HasComponent">ns=2;i=13</Reference>
  </References>
</UAObject>
<UAObject NodeId="ns=2;i=10" BrowseName="secondScrewdriver">
  <DisplayName>secondScrewdriver</DisplayName>
  <Description></Description>
  <References>
    <Reference ReferenceType="HasProperty">ns=2;i=11</Reference>
    <Reference ReferenceType="HasComponent" IsForward="false">ns=2;i=1</Reference>
    <Reference ReferenceType="HasTypeDefinition">ns=2;i=19</Reference>

```

```

    <Reference ReferenceType="ns=1;i=4001">ns=2;i=18</Reference>
    <Reference ReferenceType="ns=1;i=4001">ns=2;i=18</Reference>
    <Reference ReferenceType="HasComponent">ns=2;i=15</Reference>
  </References>
</UAObject>
<UAObject NodeId="ns=2;i=13" BrowseName="EnergySupply" ParentNodeId="ns=2;i=8">
  <DisplayName>EnergySupply</DisplayName>
  <Description></Description>
  <References>
    <Reference ReferenceType="HasProperty">ns=2;i=14</Reference>
    <Reference ReferenceType="HasComponent" IsForward="false">ns=2;i=8</Reference>
    <Reference ReferenceType="HasTypeDefinition">ns=2;i=17</Reference>
  </References>
</UAObject>
<UAObject NodeId="ns=2;i=15" BrowseName="EnergySupply" ParentNodeId="ns=2;i=10">
  <DisplayName>EnergySupply</DisplayName>
  <Description></Description>
  <References>
    <Reference ReferenceType="HasProperty">ns=2;i=16</Reference>
    <Reference ReferenceType="HasComponent" IsForward="false">ns=2;i=10</Reference>
    <Reference ReferenceType="HasTypeDefinition">ns=2;i=17</Reference>
  </References>
</UAObject>
<UAObject NodeId="ns=2;i=20" BrowseName="EnergySupply" ParentNodeId="ns=2;i=19">
  <DisplayName>EnergySupply</DisplayName>
  <Description></Description>
  <References>
    <Reference ReferenceType="HasProperty">ns=2;i=21</Reference>
    <Reference ReferenceType="HasComponent" IsForward="false">ns=2;i=19</Reference>
    <Reference ReferenceType="HasTypeDefinition">ns=2;i=17</Reference>
  </References>
</UAObject>
<UAObject NodeId="ns=2;i=22" BrowseName="Topology.aml">
  <DisplayName>Topology.aml</DisplayName>
  <Description></Description>
  <References>
    <Reference ReferenceType="HasTypeDefinition">ns=1;i=1005</Reference>
    <Reference ReferenceType="HasProperty">ns=2;i=23</Reference>
    <Reference ReferenceType="HasProperty">ns=2;i=24</Reference>
    <Reference ReferenceType="HasComponent">ns=2;i=25</Reference>
    <Reference ReferenceType="HasComponent">ns=2;i=26</Reference>
    <Reference ReferenceType="HasComponent">ns=2;i=27</Reference>
    <Reference ReferenceType="HasComponent">ns=2;i=28</Reference>
    <Reference ReferenceType="Organizes" IsForward="false">ns=1;i=5006</Reference>
  </References>
</UAObject>
<UAObject NodeId="ns=2;i=25" BrowseName="InstanceHierarchies"
ParentNodeId="ns=2;i=22">
  <DisplayName>InstanceHierarchies</DisplayName>
  <References>
    <Reference ReferenceType="HasComponent" IsForward="false">ns=2;i=22</Reference>
    <Reference ReferenceType="HasTypeDefinition">i=61</Reference>
    <Reference ReferenceType="HasComponent">ns=2;i=1</Reference>
  </References>
</UAObject>
<UAObject NodeId="ns=2;i=26" BrowseName="SystemUnitClassLibs"
ParentNodeId="ns=2;i=22">
  <DisplayName>SystemUnitClassLibs</DisplayName>
  <References>
    <Reference ReferenceType="HasComponent" IsForward="false">ns=2;i=22</Reference>
    <Reference ReferenceType="HasTypeDefinition">i=61</Reference>

```

```

    <Reference ReferenceType="HasComponent">ns=2;i=6</Reference>
  </References>
</UAObject>
<UAObject NodeId="ns=2;i=27" BrowseName="RoleClassLibs" ParentNodeId="ns=2;i=22">
  <DisplayName>RoleClassLibs</DisplayName>
  <References>
    <Reference ReferenceType="HasComponent" IsForward="false">ns=2;i=22</Reference>
    <Reference ReferenceType="HasTypeDefinition">i=61</Reference>
    <Reference ReferenceType="HasComponent">ns=2;i=4</Reference>
  </References>
</UAObject>
<UAObject NodeId="ns=2;i=28" BrowseName="InterfaceClassLibs"
ParentNodeId="ns=2;i=22">
  <DisplayName>InterfaceClassLibs</DisplayName>
  <References>
    <Reference ReferenceType="HasComponent" IsForward="false">ns=2;i=22</Reference>
    <Reference ReferenceType="HasTypeDefinition">i=61</Reference>
    <Reference ReferenceType="HasComponent">ns=2;i=2</Reference>
  </References>
</UAObject>
<UAVariable NodeId="ns=2;i=3" BrowseName="Version" ParentNodeId="ns=2;i=2"
DataType="String">
  <DisplayName>Version</DisplayName>
  <References>
    <Reference ReferenceType="HasTypeDefinition">i=68</Reference>
  </References>
  <Value>
    <String xmlns="http://opcfoundation.org/UA/2008/02/Types.xsd">1.0</String>
  </Value>
</UAVariable>
<UAVariable NodeId="ns=2;i=5" BrowseName="Version" ParentNodeId="ns=2;i=4"
DataType="String">
  <DisplayName>Version</DisplayName>
  <References>
    <Reference ReferenceType="HasTypeDefinition">i=68</Reference>
  </References>
  <Value>
    <String xmlns="http://opcfoundation.org/UA/2008/02/Types.xsd">1.0</String>
  </Value>
</UAVariable>
<UAVariable NodeId="ns=2;i=7" BrowseName="Version" ParentNodeId="ns=2;i=6"
DataType="String">
  <DisplayName>Version</DisplayName>
  <References>
    <Reference ReferenceType="HasTypeDefinition">i=68</Reference>
  </References>
  <Value>
    <String xmlns="http://opcfoundation.org/UA/2008/02/Types.xsd">1.0</String>
  </Value>
</UAVariable>
<UAVariable NodeId="ns=2;i=9" BrowseName="ID" ParentNodeId="ns=2;i=8"
DataType="String">
  <DisplayName>ID</DisplayName>
  <References>
    <Reference ReferenceType="HasTypeDefinition">i=68</Reference>
  </References>
  <Value>
    <String xmlns="http://opcfoundation.org/UA/2008/02/Types.xsd">{788eb291-f103-4fdc-
aba0-4893b599f556}</String>
  </Value>
</UAVariable>

```

```

<UAVariable NodeId="ns=2;i=11" BrowseName="ID" ParentNodeId="ns=2;i=10"
DataType="String">
  <DisplayName>ID</DisplayName>
  <References>
    <Reference ReferenceType="HasTypeDefinition">i=68</Reference>
  </References>
  <Value>
    <String xmlns="http://opcfoundation.org/UA/2008/02/Types.xsd">{19dcf818-4716-4fc1-
a85f-28e1938c4c3a}</String>
  </Value>
</UAVariable>
<UAVariable NodeId="ns=2;i=12" BrowseName="New Attribute" ParentNodeId="ns=2;i=8"
DataType="String" AccessLevel="15" UserAccessLevel="15">
  <DisplayName>New Attribute</DisplayName>
  <Description></Description>
  <References>
    <Reference ReferenceType="HasTypeDefinition">i=63</Reference>
    <Reference ReferenceType="HasComponent" IsForward="false">ns=2;i=8</Reference>
  </References>
  <Value>
    <String xmlns="http://opcfoundation.org/UA/2008/02/Types.xsd" />
  </Value>
</UAVariable>
<UAVariable NodeId="ns=2;i=14" BrowseName="ID" ParentNodeId="ns=2;i=13"
DataType="String">
  <DisplayName>ID</DisplayName>
  <References>
    <Reference ReferenceType="HasTypeDefinition">i=68</Reference>
  </References>
  <Value>
    <String xmlns="http://opcfoundation.org/UA/2008/02/Types.xsd">{5f535d4c-dd46-4c1c-
898c-4e58419048b6}</String>
  </Value>
</UAVariable>
<UAVariable NodeId="ns=2;i=16" BrowseName="ID" ParentNodeId="ns=2;i=15"
DataType="String">
  <DisplayName>ID</DisplayName>
  <References>
    <Reference ReferenceType="HasTypeDefinition">i=68</Reference>
  </References>
  <Value>
    <String xmlns="http://opcfoundation.org/UA/2008/02/Types.xsd">50e10905-ac18-413c-
afab-ad8ed1569fff</String>
  </Value>
</UAVariable>
<UAVariable NodeId="ns=2;i=21" BrowseName="ID" ParentNodeId="ns=2;i=20"
DataType="String">
  <DisplayName>ID</DisplayName>
  <References>
    <Reference ReferenceType="HasTypeDefinition">i=68</Reference>
  </References>
  <Value>
    <String xmlns="http://opcfoundation.org/UA/2008/02/Types.xsd">dd0e0dfe-10f8-4068-
845b-9c29699ac79b</String>
  </Value>
</UAVariable>
<UAVariable NodeId="ns=2;i=23" BrowseName="FileName" ParentNodeId="ns=2;i=22"
DataType="String">
  <DisplayName>FileName</DisplayName>
  <References>
    <Reference ReferenceType="HasTypeDefinition">i=68</Reference>

```

```

</References>
<Value>
  <String xmlns="http://opcfoundation.org/UA/2008/02/Types.xsd">Topology.aml</String>
</Value>
</UAVariable>
<UAVariable NodeId="ns=2;i=24" BrowseName="CAEXSchemaVersion"
ParentNodeId="ns=2;i=22" DataType="String">
  <DisplayName>CAEXSchemaVersion</DisplayName>
  <References>
    <Reference ReferenceType="HasTypeDefinition">i=68</Reference>
  </References>
  <Value>
    <String xmlns="http://opcfoundation.org/UA/2008/02/Types.xsd">2.15</String>
  </Value>
</UAVariable>
<UAObjectType NodeId="ns=2;i=17" BrowseName="Energy">
  <DisplayName>Energy</DisplayName>
  <Description></Description>
  <References>
    <Reference ReferenceType="ns=1;i=4002" IsForward="false">ns=2;i=2</Reference>
    <Reference ReferenceType="HasSubtype" IsForward="false">ns=1;i=1001</Reference>
  </References>
</UAObjectType>
<UAObjectType NodeId="ns=2;i=18" BrowseName="Tool">
  <DisplayName>Tool</DisplayName>
  <Description></Description>
  <References>
    <Reference ReferenceType="ns=1;i=4002" IsForward="false">ns=2;i=4</Reference>
    <Reference ReferenceType="HasSubtype" IsForward="false">ns=1;i=1001</Reference>
  </References>
</UAObjectType>
<UAObjectType NodeId="ns=2;i=19" BrowseName="ElectricScrewdriver">
  <DisplayName>ElectricScrewdriver</DisplayName>
  <Description></Description>
  <References>
    <Reference ReferenceType="ns=1;i=4002" IsForward="false">ns=2;i=6</Reference>
    <Reference ReferenceType="HasSubtype" IsForward="false">ns=1;i=1001</Reference>
    <Reference ReferenceType="ns=1;i=4001">ns=2;i=18</Reference>
    <Reference ReferenceType="HasComponent">ns=2;i=20</Reference>
  </References>
</UAObjectType>
</UANodeSet>

```

Figure 22: OPC UA XML text

Annex B (informative): Bibliography

- [1] Robert Henssen, Miriam Schleipen: Interoperability between OPC-UA and AutomationML. Disruptive Innovation in Manufacturing Engineering towards the 4th Industrial Revolution, Proceedings of the 8th International CIRP Conference on Digital Enterprise Technology - DET 2014 mit CD-ROM, Hrsg.: Wilhelm Bauer, Carmen Constantinescu, Olaf Sauer, Paul Maropoulos, Jody Muelaner; Fraunhofer IAO, Stuttgart; 2014, Sprache: Englisch, Fraunhofer Verlag, ISBN 978-3-8396-0697-1, 2014. Full-text: Procedia CIRP, Volume 25, 2014, Pages 297–304, <http://www.sciencedirect.com/science/article/pii/S2212827114010737>, 2014.