VDMA 40001-1

ICS 25.020; 35.240.50

Replaces
VDMA 40001-1:2022-05

OPC UA for Machinery – Part 1: Basic Building Blocks

OPC UA for Machinery –
Teil 1: Basic Building Blocks

VDMA 40001-1:2023-08 is identical with OPC 40001-1 (Release 1.03.0)

Document comprises 70 pages

VDMA

Contents

Figures

Tables

OPC Foundation / VDMA

AGREEMENT OF USE

COPYRIGHT RESTRICTIONS

- This document is provided "as is" by the OPC Foundation and VDMA.
- Right of use for this specification is restricted to this specification and does not grant rights of use for referred documents.
- Right of use for this specification will be granted without cost.
- This 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 VDMA 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 VDMA 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 this specification.
- 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 VDMA.

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OPC or VDMA specifications may require use of an invention covered by patent rights. OPC Foundation or VDMA shall not be responsible for identifying patents for which a license may be required by any OPC or VDMA specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OPC or VDMA 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 FOUDATION NOR VDMA 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 VDMA 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 VDMA 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 VDMA 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.

Foreword

Mechanical engineering is a broad-based industry, which is mainly associated with machines such as machine tools, woodworking machines or robots. Many other products such as measuring and testing equipment are also relevant to this field.

Since the information models in this document are intended to apply not only to machines (see in ISO 12100:2010, [1]), but to all other applications and products in the entire machinery industry, each case cannot be represented individually. Therefore, the term "machine" is used uniformly for all in this document.

Compared with previous versions, the following changes have been made:

Version	Changes
OPC 40001-1 1.00 (identical with VDMA 40001-1:2020-11)	Initial release
OPC 40001-1 1.01 (identical with VDMA 40001-1:2021-04)	Added Use-Cases “Component Identification” and “Finding all Components of a Machine”.
OPC 40001-1 1.02 (identical with VDMA 40001-1:2022-05)	Added Use-Case “Machine Monitoring” with the respective Building Blocks “Machinery Item State” and “Machinery Operation Mode”.
OPC 40001-1 1.03 (identical with VDMA 40001-1:2023-08)	Added Use Case “Preventive Maintenance” and Building Blocks “Operation Counter and Lifetime Counter. Extended Example in Annex C for KPI calculations.

This specification was created by a joint working group of the OPC Foundation and VDMA.

OPC Foundation

OPC is the interoperability standard for the secure and reliable exchange of data and information in the industrial automation space and in other industries. It is platform independent and ensures the seamless flow of information among devices from multiple vendors. The OPC Foundation is responsible for the development and maintenance of this standard.

OPC UA is a platform independent service-oriented architecture that integrates all the functionality of the individual OPC Classic specifications into one extensible framework. This multi-layered approach accomplishes the original design specification goals of:

– Platform independence: from an embedded microcontroller to cloud-based infrastructure

– Secure: encryption, authentication, authorization and auditing

– Extensible: ability to add new features including transports without affecting existing applications

– Comprehensive information modelling capabilities: for defining any model from simple to complex

VDMA

The VDMA is Europe’s largest industry association with over 3300 member companies of the mechanical engineering industry. These companies integrate the latest technologies in products and processes. VDMA was founded in November 1892 and is the most important voice for the mechanical engineering industry today. With the headquarters located in Frankfurt, it represents the issues of the mechanical and plant engineering sector in Germany and Europe. The standard OPC UA has established itself in this industry sector. The VDMA defines OPC UA Companion Specifications for various sectors of the mechanical engineering industry, with more than 450 companies involved. Consequently, one of the main tasks is to harmonise and create consistency.

1 Scope

The OPC UA for Machinery specification contains various building blocks for Machinery that allow to address use cases across different types of machines and components of machines defined in various companion specifications.

The content of this specification is applicable for any piece of equipment or parts of equipment that converts energy (e.g., electricity, steam, gas, human power, pressure) to mechanical movements, heat, electrical signals, pressure etc. to do a particular task in the mechanical engineering industry. This includes for example:

a) Different types of Machines (see ISO 12100:2010, [1]), e.g. machine tools, injection moulding machines, woodworking machines, packaging machinery

b) Partly completed machines, e.g. robotic systems

c) Accessory and auxiliary equipment, e.g. interchangeable equipment, load-carrying equipment

d) Devices and modules for the process industry, e.g. ovens, power systems

e) Measuring, analysis and testing equipment, e.g. machine vision systems

f) Control systems

g) The environment with which entities are energetically and/or communicatively connected

h) Installations consisting of multiple entities

This version contains building blocks for

– Machine Identification and Nameplate (see section 8)

– Finding all Machines in a Server (see section 9)

– Component Identification and Nameplate (see section 10)

– Finding all identifiable Components of a Machine (see section 11)

– *MachineryItemState* (see section 12)

– *MachineryOperationMode* (see section 13)

NOTE 1 In the context of this document, the term "machine" is used throughout the document, regardless of the application.

NOTE 2 The defined building blocks can also be used out of the context of mechanical engineering and for other applications as it is considered appropriate.

2 Normative References

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

OPC 10000-1, *OPC Unified Architecture - Part 1: Overview and Concepts*

http://www.opcfoundation.org/UA/Part1/

OPC 10000-2, *OPC Unified Architecture - Part 2: Security Model*

http://www.opcfoundation.org/UA/Part2/

OPC 10000-3, *OPC Unified Architecture - Part 3: Address Space Model*

http://www.opcfoundation.org/UA/Part3/

OPC 10000-4, *OPC Unified Architecture - Part 4: Services*

http://www.opcfoundation.org/UA/Part4/

OPC 10000-5, *OPC Unified Architecture - Part 5: Information Model*

http://www.opcfoundation.org/UA/Part5/

OPC 10000-6, *OPC Unified Architecture - Part 6: Mappings*

http://www.opcfoundation.org/UA/Part6/

OPC 10000-7, *OPC Unified Architecture - Part 7: Profiles*

http://www.opcfoundation.org/UA/Part7/

OPC 10000-16, *OPC Unified Architecture - Part 16: State Machines*

http://www.opcfoundation.org/UA/Part16/

OPC 10000-100, *OPC Unified Architecture - Part 100: Devices*

http://www.opcfoundation.org/UA/Part100/

3 Terms, Definitions and Conventions

3.1 Overview

It is assumed that basic concepts of OPC UA information modelling are understood in this document. This document will use these concepts to describe the OPC UA for Machinery Information Model. For the purposes of this document, the terms and definitions given in OPC 10000-1, OPC 10000-3, OPC 10000-4, OPC 10000-5, OPC 10000-7, and OPC 10000-100 as well as the following apply.

Note that OPC UA terms and terms defined in this specification are *italicized* in the specification.

3.2 OPC UA for Machinery Terms

MachineryItem
Machine or component of a *Machine*

3.3 Abbreviated Terms

e.g.	for example
	(OL: exempli gratia)
ERP	Enterprise-Resource-Planning
HMI	Human-Machine Interface
HTTP	Hypertext Transfer Protocol
i.e.	that is to say
	(OL: id est)
IP	Internet Protocol
ISO	International Organization for Standardization
KPI	Key Performance Indicator
MES	Manufacturing Execution System
OL	Original Language
OPC UA	Open Platform Communications Unified Architecture
PLC	Programmable Logical Controller
PMS	Production Management System
TCP	Transmission Control Protocol
UML	Unified Modeling Language
URI	Uniform Resource Identifier
VDMA	German Mechanical Engineering Industry Association
	(OL: Verband Deutscher Maschinen- und Anlagenbau)
XML	Extensible Markup Language

3.4 Conventions used in this Document

3.4.1 Conventions for Node Descriptions

3.4.1.1 Node Definitions

Node definitions are specified using tables (see Table 2).

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.

– 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 OPC 10000-3). 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 OPC 10000-3) and the *ArrayDimensions* is set to null or is omitted. Examples are given in Table 1.

Table 1 – Examples of DataTypes

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

– The TypeDefinition is specified for *Objects* and *Variables*.

– The TypeDefinition column specifies a symbolic name for a *NodeId*, i.e. the specified *Node* points with a *HasTypeDefinition Reference* to the corresponding *Node*.

– 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.

Note that if a symbolic name of a different namespace is used, it is prefixed by the *NamespaceIndex* (see 3.4.2.2).

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 part of this document points to their definition.

Table 2 illustrates the table. 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.

Each *Type Node* or well-known *Instance Node* defined shall have one or more *ConformanceUnits* defined in 16.1 that require the *Node* to be in the *AddressSpace*.

The relations between *Nodes* and *ConformanceUnits* are defined at the end of the tables defining *Nodes*, one row per *ConformanceUnit*. The *ConformanceUnits* are reflected in the *Category* element for the *Node* definition in the *UANodeSet* (see OPC 10000-6).

The list of *ConformanceUnits* in the *UANodeSet* allows *Servers* to optimize resource consumption by using a list of supported *ConformanceUnits* to select a subset of the *Nodes* in an *Information Model*.

When a *Node* is selected in this way, all dependencies implied by the *References* are also selected.

Dependencies exist if the *Node* is the source of *HasTypeDefinition*, *HasInterface*, *HasAddIn* or any *HierarchicalReference*. Dependencies also exist if the *Node* is the target of a *HasSubtype Reference*. For *Variables* and *VariableTypes*, the value of the *DataType Attribute* is a dependency. For *DataType Nodes*, any *DataTypes* referenced in the *DataTypeDefinition Attribute* are also dependencies.

For additional details see OPC 10000-5.

Table 2 – Type Definition Table

Attribute	**Value**				
Attribute name	Attribute value. If it is an optional Attribute that is not set "--" will be used.				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
ReferenceType name	*NodeClass* of the Target*Node*.	*BrowseName* of the target *Node*.	*DataType* of the referenced *Node*, only applicable for *Variables*.	*TypeDefinition* of the referenced *Node*, only applicable for *Variables* and *Objects*.	Additional characteristics of the *TargetNode* such as the *ModellingRule* or *AccessLevel*.
NOTE Notes referencing footnotes of the table content.					
Conformance Units					
Name of ConformanceUnit, one row per ConformanceUnit					

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

The Other column defines additional characteristics of the Node. Examples of characteristics that can appear in this column are show in Table 3.

Table 3 – Examples of Other Characteristics

Name	**Short Name**	**Description**
0:Mandatory	M	The Node has the Mandatory ModellingRule.
0:Optional	O	The Node has the Optional ModellingRule.
0:MandatoryPlaceholder	MP	The Node has the MandatoryPlaceholder ModellingRule.
0:OptionalPlaceholder	OP	The Node has the OptionalPlaceholder ModellingRule.
ReadOnly	RO	The Node AccessLevel has the CurrentRead bit set but not the CurrentWrite bit.
ReadWrite	RW	The Node AccessLevel has the CurrentRead and CurrentWrite bits set.
WriteOnly	WO	The Node AccessLevel has the CurrentWrite bit set but not the CurrentRead bit.

If multiple characteristics are defined they are separated by commas. The name or the short name may be used.

3.4.1.2 Additional References

To provide information about additional *References*, the format as shown in Table 4 is used.

Table 4 – <some>Type Additional References

SourceBrowsePath	Reference Type	Is Forward	TargetBrowsePath
SourceBrowsePath is always relative to the *TypeDefinition*. Multiple elements are defined as separate rows of a nested table.	*ReferenceType* name	True = forward *Reference*.	TargetBrowsePath points to another *Node*, which can be a well-known instance or a *TypeDefinition*. You can use *BrowsePaths* here as well, which is either relative to the *TypeDefinition* or absolute. If absolute, the first entry needs to refer to a type or well-known instance, uniquely identified within a namespace by the *BrowseName*.

References can be to any other *Node*.

3.4.1.3 Additional Sub-components

To provide information about sub-components, the format as shown in Table 5 is used.

Table 5 – <some>Type Additional Subcomponents

BrowsePath	References	NodeClass	BrowseName	DataType	TypeDefinition	Others
BrowsePath is always relative to the *TypeDefinition*. Multiple elements are defined as separate rows of a nested table	NOTE Same as for Table 2					

3.4.1.4 Additional Attribute Values

The type definition table provides columns to specify the values for required *Node Attributes* for *InstanceDeclarations*. To provide information about additional *Attributes*, the format as shown in Table 6 is used.

Table 6 – <some>Type Attribute Values for Child Nodes

BrowsePath	<Attribute name> Attribute
BrowsePath is always relative to the *TypeDefinition*. Multiple elements are defined as separate rows of a nested table	The values of attributes are converted to text by adapting the reversible JSON encoding rules defined in OPC 10000-6. If the JSON encoding of a value is a JSON string or a JSON number then that value is entered in the value field. Double quotes are not included. If the DataType includes a NamespaceIndex (QualifiedNames, NodeIds or ExpandedNodeIds) then the notation used for BrowseNames is used. If the value is an Enumeration the name of the enumeration value is entered. If the value is a Structure then a sequence of name and value pairs is entered. Each pair is followed by a newline. The name is followed by a colon. The names are the names of the fields in the DataTypeDefinition. If the value is an array of non-structures then a sequence of values is entered where each value is followed by a newline. If the value is an array of Structures or a Structure with fields that are arrays or with nested Structures then the complete JSON array or JSON object is entered. Double quotes are not included.

There can be multiple columns to define more than one *Attribute*.

3.4.2 NodeIds and BrowseNames

3.4.2.1 NodeIds

The *NodeIds* of all *Nodes* described in this standard are only symbolic names. Annex A defines the actual *NodeIds*.

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 *NamespaceUri* for all *NodeIds* defined in this document is defined in Annex A. The *NamespaceIndex* for this *NamespaceUri* is vendor-specific and depends on the position of the *NamespaceUri* in the server namespace table.

Note that this document not only defines concrete *Nodes*, but also requires that some *Nodes* shall be generated, for example one for each *Session* running on the *Server*. 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 in this document, because they are not defined by this document but generated by the *Server*.

3.4.2.2 BrowseNames

The text part of the *BrowseNames* for all *Nodes* defined in this document is specified in the tables defining the *Nodes*. The *NamespaceUri* for all *BrowseNames* defined in this document is defined in Annex A.

For *InstanceDeclarations* of *NodeClass Object* and *Variable* that are placeholders (*OptionalPlaceholder* and *MandatoryPlaceholder ModellingRule*), the *BrowseName* and the *DisplayName* are enclosed in angle brackets (<>) as recommended in OPC 10000-3.

If the *BrowseName* is not defined by this document, a namespace index prefix like '0:EngineeringUnits' or '2:DeviceRevision' 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 document. Table 52 provides a list of namespaces and their indexes as used in this document.

3.4.3 Common Attributes

3.4.3.1 General

The *Attributes* of *Nodes*, their *DataTypes* and descriptions are defined in OPC 10000-3. Attributes not marked as optional are mandatory and shall be provided by a *Server*. The following tables define if the *Attribute* value is defined by this specification or if it is server-specific.

For all *Nodes* specified in this specification, the *Attributes* named in Table 7 shall be set as specified in the table.

Table 7 – Common Node Attributes

Attribute	Value
DisplayName	The *DisplayName* is a *LocalizedText*. Each server shall provide the *DisplayName* identical to the *BrowseName* of the *Node* for the LocaleId "en". Whether the server provides translated names for other LocaleIds is server-specific.
Description	Optionally a server-specific description is provided.
NodeClass	Shall reflect the *NodeClass* of the *Node.*
NodeId	The *NodeId* is described by *BrowseNames* as defined in 3.4.2.1.
WriteMask	Optionally the *WriteMask Attribute* can be provided. If the *WriteMask Attribute* is provided, it shall set all non-server-specific *Attributes* to not writable. For example, the *Description Attribute* may be set to writable since a *Server* may provide a server-specific description for the *Node*. The *NodeId* shall not be writable, because it is defined for each *Node* in this specification.
UserWriteMask	Optionally the *UserWriteMask Attribute* can be provided. The same rules as for the *WriteMask Attribute* apply.
RolePermissions	Optionally server-specific role permissions can be provided.
UserRolePermissions	Optionally the role permissions of the current Session can be provided. The value is server-specific and depend on the *RolePermissions Attribute* (if provided) and the current *Session*.
AccessRestrictions	Optionally server-specific access restrictions can be provided.

3.4.3.2 Objects

For all *Objects* specified in this specification, the *Attributes* named in Table 8 shall be set as specified in the table. The definitions for the *Attributes* can be found in OPC 10000-3.

Table 8 – Common Object Attributes

Attribute	Value
EventNotifier	Whether the *Node* can be used to subscribe to *Events* or not is server-specific.

3.4.3.3 Variables

For all *Variables* specified in this specification, the *Attributes* named in Table 9 shall be set as specified in the table. The definitions for the *Attributes* can be found in OPC 10000-3.

Table 9 – Common Variable Attributes

Attribute	Value
MinimumSamplingInterval	Optionally, a server-specific minimum sampling interval is provided.
AccessLevel	The access level for *Variables* used for type definitions is server-specific, for all other *Variables* defined in this specification, the access level shall allow reading; other settings are server-specific.
UserAccessLevel	The value for the *UserAccessLevel Attribute* is server-specific. It is assumed that all *Variables* can be accessed by at least one user.
Value	For *Variables* used as *InstanceDeclarations,* the value is server-specific; otherwise it shall represent the value described in the text.
ArrayDimensions	If the *ValueRank* does not identify an array of a specific dimension (i.e. *ValueRank* <= 0) the *ArrayDimensions* can either be set to null or the *Attribute* is missing. This behaviour is server-specific. If the *ValueRank* specifies an array of a specific dimension (i.e. *ValueRank* > 0) then the *ArrayDimensions Attribute* shall be specified in the table defining the *Variable*.
Historizing	The value for the *Historizing Attribute* is server-specific.
AccessLevelEx	If the *AccessLevelEx Attribute* is provided, it shall have the bits 8, 9, and 10 set to 0, meaning that read and write operations on an individual *Variable* are atomic, and arrays can be partly written.

3.4.3.4 VariableTypes

For all *VariableTypes* specified in this specification, the *Attributes* named in Table 10 shall be set as specified in the table. The definitions for the *Attributes* can be found in OPC 10000-3.

Table 10 – Common VariableType Attributes

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

3.4.3.5 Methods

For all *Methods* specified in this specification, the *Attributes* named in Table 11 shall be set as specified in the table. The definitions for the *Attributes* can be found in OPC 10000-3.

Table 11 – Common Method Attributes

Attributes	Value
Executable	All *Methods* defined in this specification shall be executable (*Executable Attribute* set to “True”), unless it is defined differently in the *Method* definition.
UserExecutable	The value of the *UserExecutable Attribute* is server-specific. It is assumed that all *Methods* can be executed by at least one user.

4 General information to Machinery and OPC UA

4.1 Introduction to Machinery

4.1.1 Machinery and Mechanical Engineering

Machinery is the entirety of the products of mechanical engineering. Mechanical engineering is one of the oldest engineering sciences. It is understood as a branch of industry as well as an engineering discipline. This field of activity includes the development, construction and production of machines and machine parts. As a branch of industry, mechanical engineering originated from the craft of metalworking by blacksmiths and locksmiths. As an engineering discipline according to modern understanding, it is based on a systematic scientific reference to physics, especially mechanics.

4.1.2 Sample Industries and Products

The following section is intended to give an exemplary sketch of the vastness of mechanical and plant engineering. This includes for example the following industries and products:

– Food Processing and Packaging Machinery, e.g., bakery machines, meat processing machines and packaging machines.

– Robotics and Automation, e.g., Robots, machine vision systems and integrated assembly solutions

– Plastics and Rubber Machinery, e.g., Injection moulding machines and extrusion devices

– Metallurgy, e.g., foundry machinery, furnaces, metallurgical plants and rolling mills

– Materials Handling and Intralogistics, e.g., automated guided vehicles, industrial trucks and cranes

Other industries include the following:

Agricultural Machinery, Air Conditioning and Ventilation, Air Pollution Control, Air-handling Technology, Battery Production, Building Control and Management, Ceramic Machinery, Cleaning Systems, Compressed Air and Vacuum Technology, Construction-Equipment and Plant Engineering, Die and Mould, Drying Technology, Electrical Automation, Electronics, Micro and Nano Technologies, Engines and Systems, Fire Fighting Equipment, Fluid Power Industry, Glass Machinery, Lifts and Escalators, Machine Tools and Manufacturing Systems, Measuring and Testing Technology, Micro Technologies, Mining Industry, Photovoltaic Equipment, Power Systems, Power Transmission Engineering, Precision Tools, Printing and Paper Technology, Process Plant and Equipment, Pumps and Systems, Refrigeration and Heat Pump Technology, Security Systems, Surface Technology, Software and Digitalization, Textile Care, Fabric and Leather Technology, Textile Machinery, Valves, Waste Treatment and Recycling, Welding and Pressure Gas Equipment, Woodworking Machinery.

4.1.3 Machinery and OPC UA

Classical mechanical engineering is undergoing change. Driven by digitalization, the classical disciplines of mechanical engineering, such as

– technical mechanics

– thermodynamics or

– material technology

get extended by, for example

– automation technology and

– virtual product development.

In this context, the companies of this working group have identified OPC UA as a standardized interface for data exchange. As a result, they are working on features, use cases and information models that apply across the board to the entire mechanical engineering sector.

4.2 Introduction to OPC Unified Architecture

4.2.1 What is OPC UA?

OPC UA is an open and royalty free set of standards designed as a universal communication protocol. While there are numerous communication solutions available, OPC UA has key advantages:

– A state of art security model (see OPC 10000-2).

– A fault tolerant communication protocol.

– An information modelling framework that allows application developers to represent their data in a way that makes sense to them.

OPC UA has a broad scope which delivers for economies of scale for application developers. This means that a larger number of high-quality applications at a reasonable cost are available. When combined with semantic

models such as OPC UA for Machinery, OPC UA makes it easier for end users to access data via generic commercial applications.

The OPC UA model is scalable from small devices to ERP systems. OPC UA *Servers* process information locally and then provide that data in a consistent format to any application requesting data – ERP, MES, PMS, Maintenance Systems, HMI, Smartphone or a standard Browser, for examples. For a more complete overview see OPC 10000-1.

4.2.2 Basics of OPC UA

As an open standard, OPC UA is based on standard internet technologies, like TCP/IP, HTTP, Web Sockets.

As an extensible standard, OPC UA provides a set of *Services* (see OPC 10000-4) and a basic information model framework. This framework provides an easy manner for creating and exposing vendor defined information in a standard way. More importantly all OPC UA *Clients* are expected to be able to discover and use vendor-defined information. This means OPC UA users can benefit from the economies of scale that come with generic visualization and historian applications. This specification is an example of an OPC UA *Information Model* designed to meet the needs of developers and users.

OPC UA *Clients* can be any consumer of data from another device on the network to browser based thin clients and ERP systems. The full scope of OPC UA applications is shown in Figure 1.

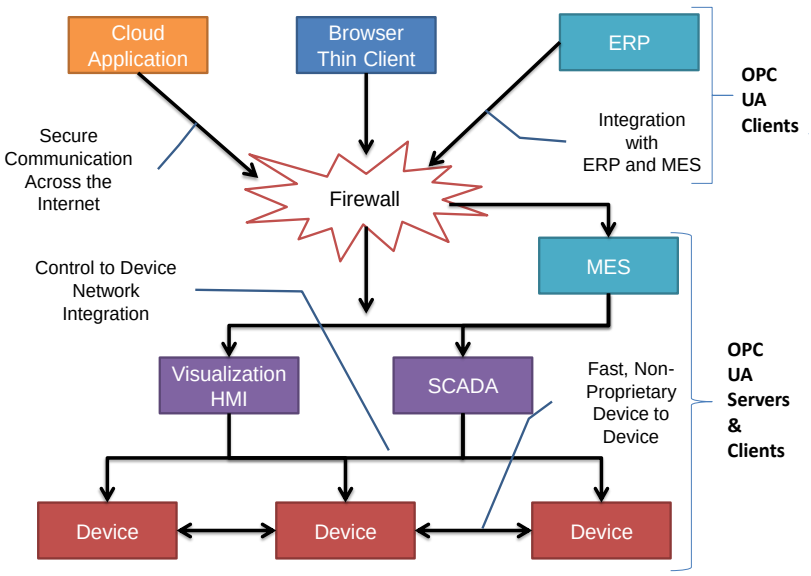

Figure 1 – The Scope of OPC UA within an Enterprise

OPC UA provides a robust and reliable communication infrastructure having mechanisms for handling lost messages, failover, heartbeat, etc. With its binary encoded data, it offers a high-performing data exchange solution. Security is built into OPC UA as security requirements become more and more important especially since environments are connected to the office network or the internet and attackers are starting to focus on automation systems.

4.2.3 Information Modelling in OPC UA

4.2.3.1 Concepts

OPC UA provides a framework that can be used to represent complex information as *Objects* in an *AddressSpace* which can be accessed with standard services. These *Objects* consist of *Nodes* connected by *References*. Different classes of *Nodes* convey different semantics. For example, a *Variable Node* represents a value that can be read or written. The *Variable Node* has an associated *DataType* that can define the actual value, such as a string, float, structure etc. It can also describe the *Variable* value as a variant. A *Method Node* represents a function that can be called. Every *Node* has a number of *Attributes* including a unique identifier

called a *NodeId* and non-localized name called as *BrowseName*. An *Object* representing a ‘Reservation’ is shown in Figure 2.

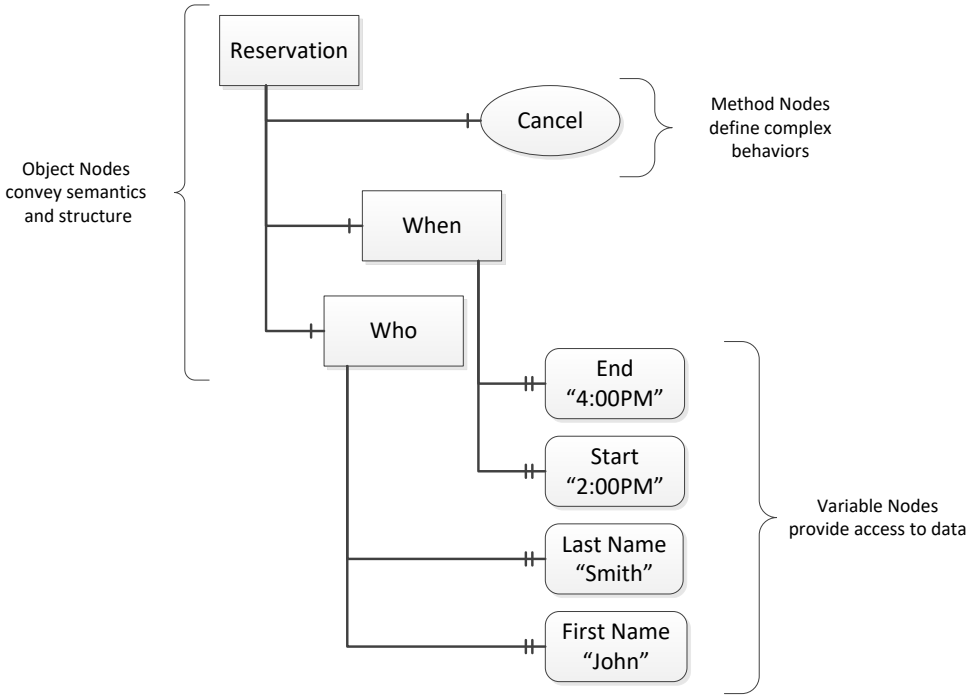

Figure 2 – A Basic Object in an OPC UA Address Space

Object and *Variable Nodes* represent instances and they always reference a *TypeDefinition* (*ObjectType* or *VariableType*) *Node* which describes their semantics and structure. Figure 3 illustrates the relationship between an instance and its *TypeDefinition*.

The type *Nodes* are templates that define all of the children that can be present in an instance of the type. In the example in Figure 3 the PersonType *ObjectType* defines two children: First Name and Last Name. All instances of PersonType are expected to have the same children with the same *BrowseNames*. Within a type the *BrowseNames* uniquely identify the children. This means *Client* applications can be designed to search for children based on the *BrowseNames* from the type instead of *NodeIds*. This eliminates the need for manual reconfiguration of systems if a *Client* uses types that multiple *Servers* implement.

OPC UA also supports the concept of subtyping. This allows a modeller to take an existing type and extend it. There are rules regarding subtyping defined in OPC 10000-3, but in general they allow the extension of a given type or the restriction of a *DataType*. For example, the modeller may decide that the existing *ObjectType* in some cases needs an additional *Variable*. The modeller can create a subtype of the *ObjectType* and add the *Variable*. A *Client* that is expecting the parent type can treat the new type as if it was of the parent type. Regarding *DataTypes*, subtypes can only restrict. If a *Variable* is defined to have a numeric value, a sub type could restrict it to a float.

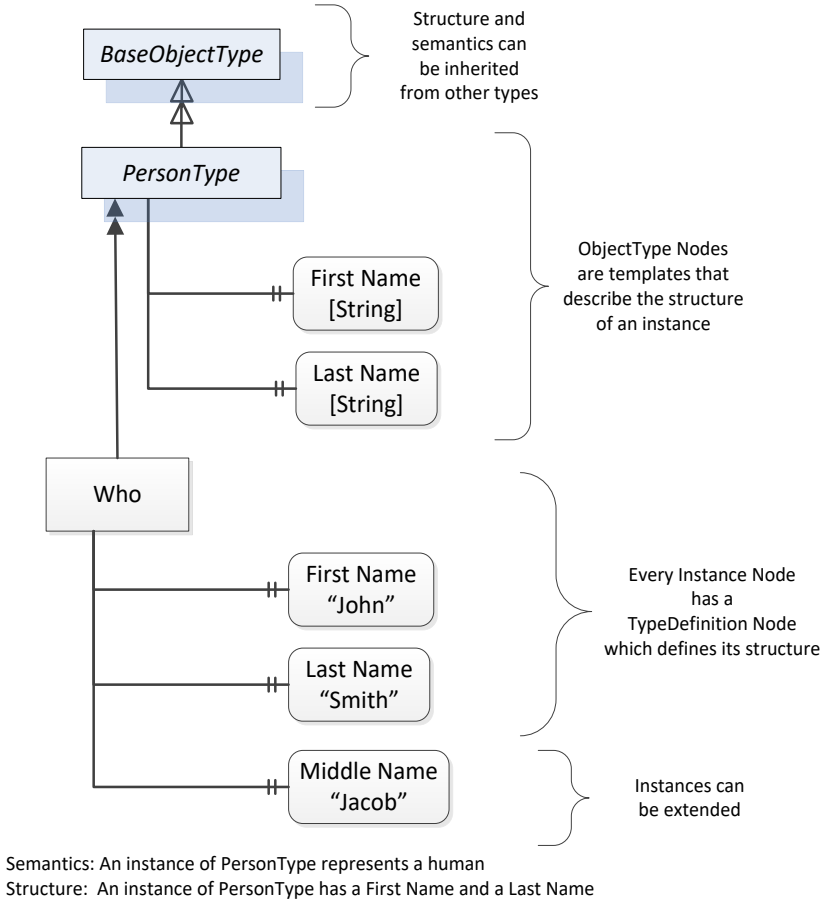

Figure 3 – The Relationship between Type Definitions and Instances

References allow *Nodes* to be connected in ways that describe their relationships. All *References* have a *ReferenceType* that specifies the semantics of the relationship. *References* can be hierarchical or non-hierarchical. Hierarchical references are used to create the structure of *Objects* and *Variables*. Non-hierarchical are used to create arbitrary associations. Applications can define their own *ReferenceType* by creating subtypes of an existing *ReferenceType*. Subtypes inherit the semantics of the parent but may add additional restrictions. Figure 4 depicts several *References,* connecting different *Objects*.

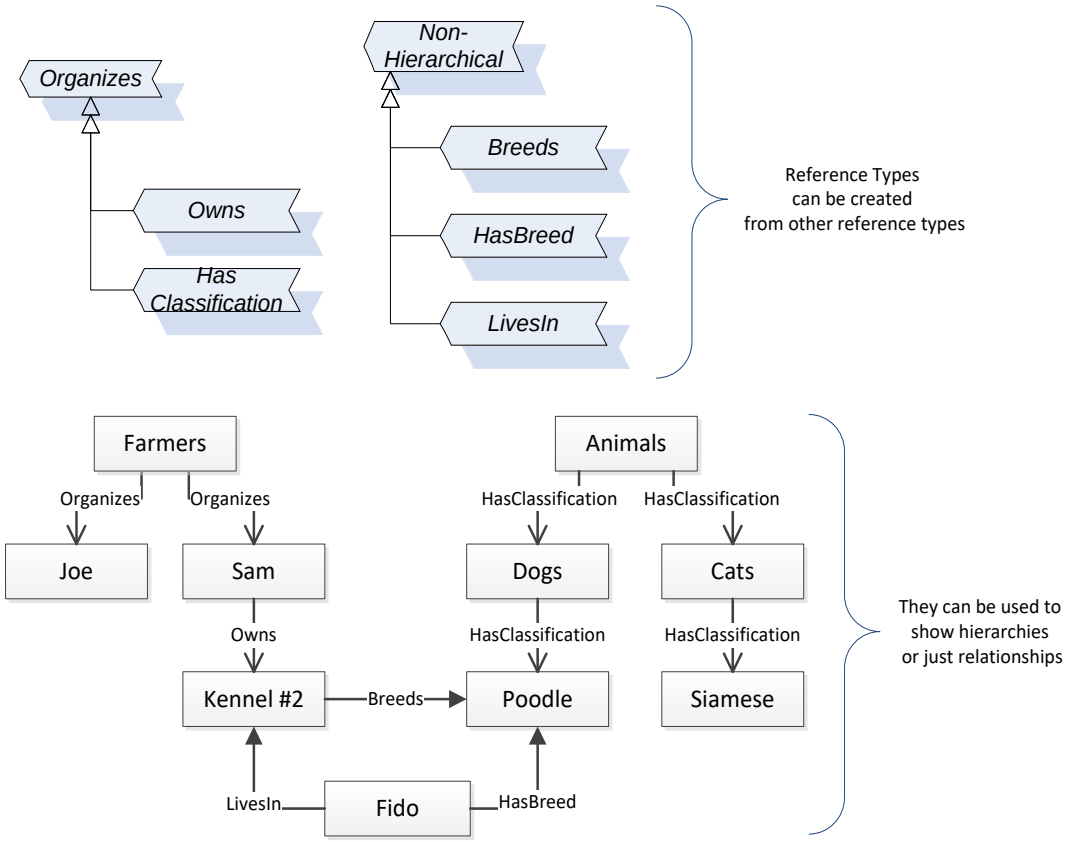

Figure 4 – Examples of References between Objects

The figures above use a notation that was developed for the OPC UA specification. The notation is summarized in Figure 5. UML representations can also be used; however, the OPC UA notation is less ambiguous because there is a direct mapping from the elements in the figures to *Nodes* in the *AddressSpace* of an OPC UA *Server*.

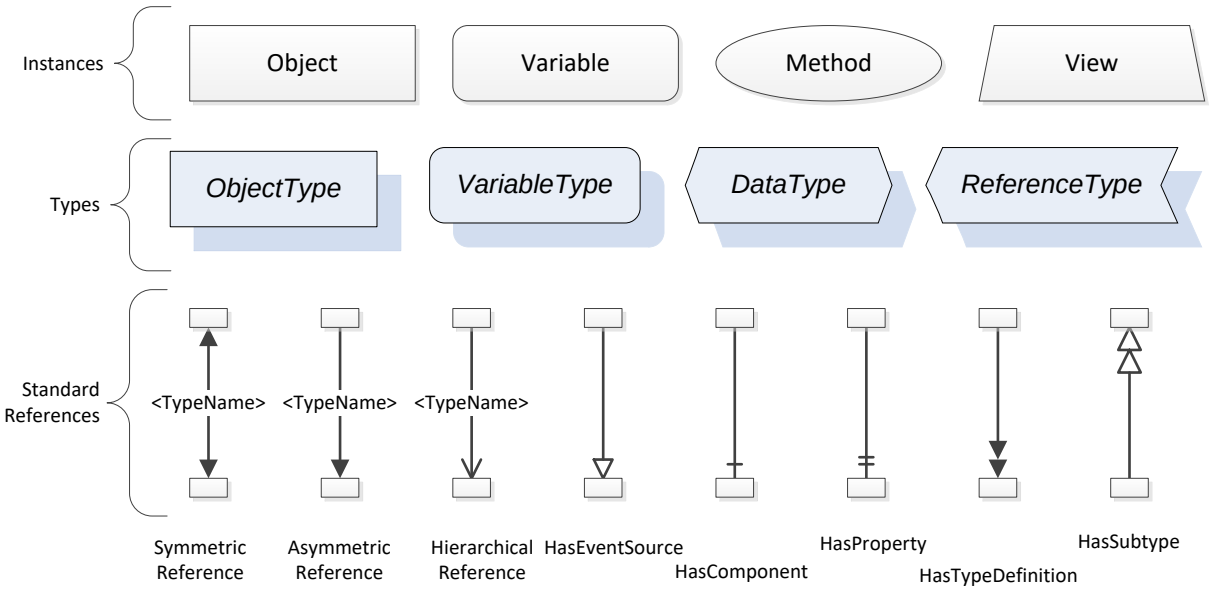

Figure 5 – The OPC UA Information Model Notation

A complete description of the different types of Nodes and References can be found in OPC 10000-3 and the base structure is described in OPC 10000-5.

OPC UA specification defines a very wide range of functionality in its basic information model. It is not required that all *Clients* or *Servers* support all functionality in the OPC UA specifications. OPC UA includes the concept of *Profiles*, which segment the functionality into testable certifiable units. This allows the definition of functional subsets (that are expected to be implemented) within a companion specification. The *Profiles* do not restrict functionality, but generate requirements for a minimum set of functionality (see OPC 10000-7).

4.2.3.2 Namespaces

OPC UA allows information from many different sources to be combined into a single coherent *AddressSpace*. Namespaces are used to make this possible by eliminating naming and id conflicts between information from different sources. Each namespace in OPC UA has a globally unique string called a NamespaceUri which identifies a naming authority and a locally unique integer called a NamespaceIndex, which is an index into the *Server's* table of *NamespaceUris*. The *NamespaceIndex* is unique only within the context of a *Session* between an OPC UA *Client* and an OPC UA *Server*- the *NamespaceIndex* can change between *Sessions* and still identify the same item even though the NamespaceUri's location in the table has changed. The *Services* defined for OPC UA use the *NamespaceIndex* to specify the Namespace for qualified values.

There are two types of structured values in OPC UA that are qualified with *NamespaceIndexes*: NodeIds and *QualifiedNames*. NodeIds are locally unique (and sometimes globally unique) identifiers for *Nodes*. The same globally unique *NodeId* can be used as the identifier in a node in many *Servers* – the node's instance data may vary but its semantic meaning is the same regardless of the *Server* it appears in. This means *Clients* can have built-in knowledge of what the data means in these *Nodes*. OPC UA *Information Models* generally define globally unique *NodeIds* for the *TypeDefinitions* defined by the *Information Model*.

QualifiedNames are non-localized names qualified with a Namespace. They are used for the *BrowseNames* of *Nodes* and allow the same names to be used by different information models without conflict. *TypeDefinitions* are not allowed to have children with duplicate *BrowseNames*; however, instances do not have that restriction.

4.2.3.3 Companion Specifications

An OPC UA companion specification for an industry specific vertical market describes an *Information Model* by defining *ObjectTypes*, *VariableTypes*, *DataTypes* and *ReferenceTypes* that represent the concepts used in the vertical market, and potentially also well-defined Objects as entry points into the AddressSpace.

5 Use Cases

This specification provides building blocks for various use cases. Other specifications or vendor-specific information models can pick the building blocks for specific use cases they want to support.

5.1 Machine Identification and Nameplate

The user would like to uniquely identify machines, potentially across various OPC UA Servers or aggregating OPC UA Servers. The user wants to get standardized information about the machine, like manufacturer or serial number, and set user-specific information in order to simplify the usage of the machine.

That leads to the requirements:

– A machine shall be globally uniquely identified (see section 8, *ProductInstanceUri*).

– Information about the machine, like manufacturer or serial number, can be accessed (see section 8, *IMachineVendorNameplateType*).

– Application-specific information about a machine can be set by an OPC UA Client (see section 8, *IMachineTagNameplateType*).

5.2 Finding all Machines in a Server

The user would like to easily find all machines managed by an OPC UA server.

That leads to the requirement:

– All machines shall be easy to find in an OPC UA Server (see section 9, *Machines Object*).

5.3 Component Identification and Nameplate

The user would like to identify components of a machine. The user wants to get standardized information about the component, like manufacturer or serial number, and set user-specific information in order to simplify the usage of the component.

That leads to the requirements:

– Information about the component, like manufacturer or serial number, can be accessed (see section 10.2, *MachineryComponentIdentificationType*).

– Application-specific information about a component can be set by an OPC UA Client (see section 10.2, *MachineryComponentIdentificationType*).

5.4 Finding all Components of a Machine

The user would like to easily find all components related to a specific machine.

That leads to the requirement:

– All components of a machine shall be easy to find in an OPC UA Server (see section 11.2, *MachineComponentsType*).

5.5 Machine Monitoring

The user would like to monitor information about a specific *MachineryItem*, like the state or operation mode, for example to get a quick overview over the current state and bottlenecks (localization of errors), to recognize trends or to determine the relevant times (productive time, standby time, etc.) for subsequent KPI calculations (e.g. for calculating reliability and availability). Further information for the KPI calculation might be necessary.

That leads to the requirement:

– The state of a *MachineryItem* can be accessed (see section 12).

– The operation mode of a *MachineryItem* can be accessed (see section 13).

Note: This version of the specification defines some base information for this use case. There is more information that could be defined for this use case, which might be addressed by later versions of this specification, domain-specific companion specifications or vendors.

In Annex C an example is given, on how the state and operation mode can be used as base for KPI calculations.

5.6 Preventive Maintenance

The user would like to monitor how long a *MachineryItem* is powered on and doing an activity and wants to monitor the expected remaining lifetime of a *MachineryItem* or other aspects of a machine (like the remaining time a software licence is valid).

That leads to the requirements:

– The total time a *MachineryItem* is turned on and the total time an activity is done can be accessed (see section 14).

– The remaining estimated lifetime of a *MachineryItem* or other aspects of a machine can be accessed (see section 15).

6 Machinery Information Model overview

6.1 General Idea – Definition of Building Blocks

This specification defines several building blocks for various use cases in the context of machinery. The specification uses the *AddIn* concept defined in OPC 10000-3, in order to allow companion specifications and vendors to easily apply individual building blocks.

This is exemplified in Figure 6. On the right side of the figure, you can see *ObjectTypes* defining specific functionality like Identification. This includes the definition of a default *BrowseName*. On the left side you see an example of how such a building block is used. This type could use other building blocks as well.

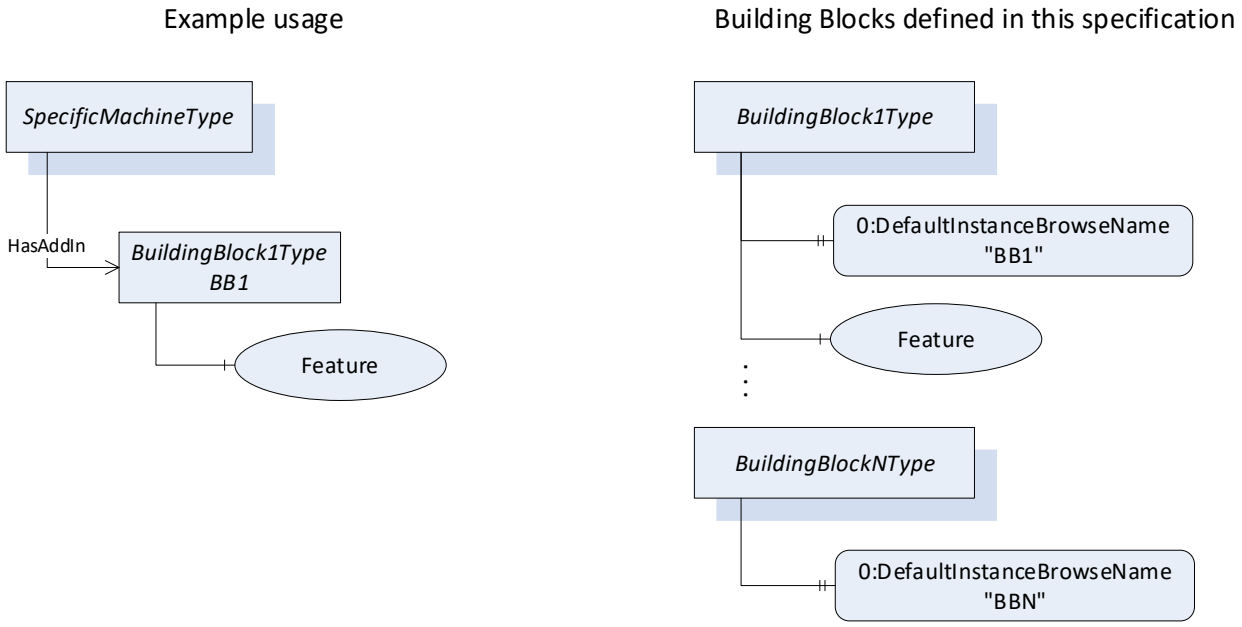

Figure 6 – Concept of Building Blocks

6.2 Overview of the Building Blocks

This version of the specification defines

– a building block for *Machine* Identification and Nameplate (see section 8) defined as *AddIn*

– capabilities to find all *Machines* in a *Server* (see section 9) by defining a standardized entry point

– a building block for component Identification and Nameplate (see section 10) defined as *AddIn*

– a building block to find all identifiable components of a *Machine* (see section 11) defined as *AddIn*

– a building block to represent the state of a *MachineryItem* (see section 12) defined as *AddIn*

– a building block to represent the operation mode of a *MachineryItem* (see section 13) defined as *AddIn*

– a building block for operation counters (see section 14) defined as *AddIn*

– a building block for lifetime counters (see section 15) defined as *AddIn*

6.3 Organization of Building Blocks

The building blocks defined in this specification are typically *AddIns* that can be applied to *Objects* or *ObjectTypes* representing *Machines* or components of *Machines*.

The specification intentionally does not define an *ObjectType* representing a *MachineryItem* and intentionally leaves it open how the information for the *MachineryItem* is further structured in addition to the *AddIns*. This allows a domain-specific organization of the *MachineryItem*, for example defined in a domain-specific companion specification.

It is to avoid that the *Objects* and *ObjectTypes* representing a *MachineryItem* become a large, hard to understand substructure and it is also to avoid having flat list of all the building blocks. Therefore, this specification defines an organization *Object* containing the building blocks defined in this specification. This reduces the *Nodes* on the top-level and thus simplifies accessing the information.

Each *Object* or *ObjectType* representing a *MachineryItem* supporting the *AddIns* should have an *Object* of type *FolderType* or a subtype with the *BrowseName* “MachineryBuildingBlocks” (using the Namespace of this specification), referenced with *HasComponent* or a subtype. All *AddIns* defined in this specification should be applied to this *Object*, i.e. being referenced with a *HasAddIn Reference* or a subtype from this *Object*. Because of the base characteristics of the identification and the relation to its subcomponents, those *AddIns* should be

referenced directly from the *Object* or *ObjectType* representing the *MachineryItem*, and should be referenced in addition by the *MachineryBuildingBlocks Object*.

In Figure 7, an example is given. The SpecificComponentType supports the Identification *AddIn* as well as the BuildingBlock1 and BuildingBlock2. All three are referenced from the MachineryBuildingBlocks *Object*. The Identification *AddIn* is also referenced directly from the *ObjectType*, and the other *AddIns* from some domain-specific Objects to structure the component. The SpecificMachineType supports Identification and those two additional building blocks as well, and contains a Component1 of *SpecificComponentType*. Thus, the Components *AddIn* is referenced from the *ObjectType* directly, as well as from the MachineryBuildingBlocks *Object*.

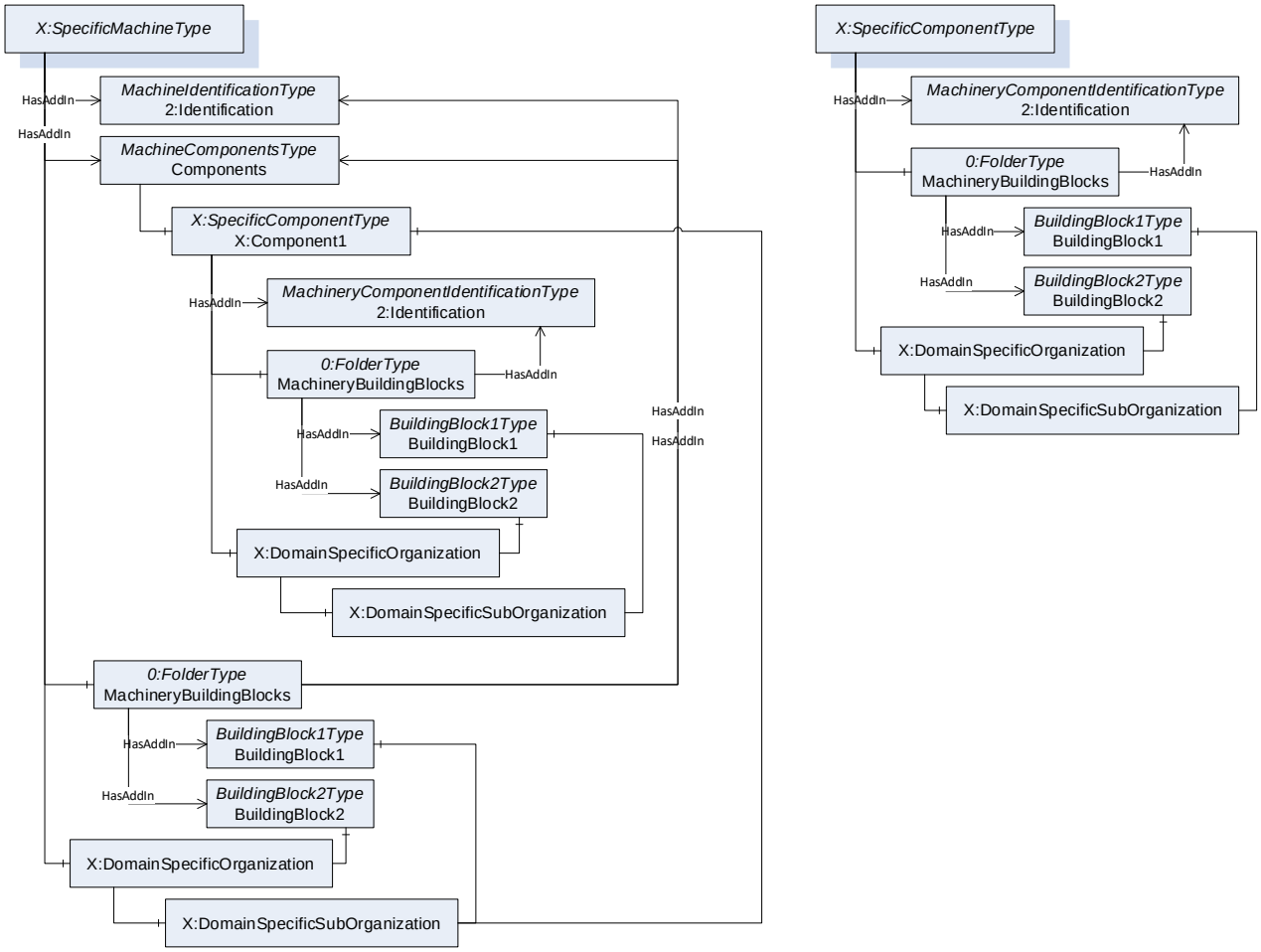

Figure 7 – Example of organization of Building Blocks

Table 12 defines, which building blocks shall or may be referenced by the MachineryBuildingBlocks folder.

Table 12 – Usage of MachineryBuildingBlocks

Building Block	Shall be in MachineryBuildingBlocks folder	May be in MachineryBuildingBlocks folder
Machine Identification and Nameplate (see 8)	-	X
Finding all Machines in a Server (see 9)	-	-
Component Identification and Nameplate (see10)	-	X
Finding all Identifiable Components of a Machine (see 11)	-	X
MachineryItemState (see 12)	X	-
MachineryOperationMode (see 13)	X	-
OperationCounters (see 14)	X	-
LifetimeCounters (see 15)	X	-

7 General Recommendations

7.1 Localization

If the text part of a value of *DataType LocalizedText*, like the Manufacturer or the Model of a Machine, is language neutral, i.e. it is the same in all languages, the locale of the *LocalizedText* shall be null or an empty string.

7.2 Optional Nodes

If the information for optional nodes (e.g. Properties) is not available and the access is read-only, the optional Node shall not be provided.

If the content of optional nodes is writable, i.e. it can be provided by end-users, system integrators, etc., it is desirable to provide the Nodes to allow the usage of them.

8 Machine Identification and Nameplate

8.1 Overview

This building block provides the capabilities to globally uniquely identify a *Machine* and have access to vendor-defined information about the *Machine* and manage user-specific information for the identification of the *Machine*. Figure 8 gives an overview. The Interface *IMachineryItemVendorNameplateType* and the *ObjectType MachineryItemIdentificationType* are generic *ObjectTypes* introduced to be used in other use cases. The *AddIn MachineIdentificationType* with the default name “2:Identification” (as defined in OPC 10000-100), is derived from the *MachineryItemIdentificationType* and thus indirectly from the *2:FunctionalGroupType* and implements the interfaces *IMachineVendorNameplateType* and *IMachineTagNameplateType*. *IMachineVendorNameplateType* is a subtype of *IMachineryItemVendorNameplateType* and thus indirectly from the *2:IVendorNameplateType* defined in OPC 10000-100. *IMachineryItemVendorNameplateType* refines the usage of the *Properties* defined in *2:IVendorNameplateType*, changes some to *Mandatory* and defines additional *Properties*. *IMachineVendorNameplateType* uses those definitions and makes another *Property* mandatory. *IMachineTagNameplateType* is a subtype of the *2:ITagNameplateType* defined in OPC 10000-100 and refines the usage of the *Properties* defined in that interface, and defines an additional *Property*.

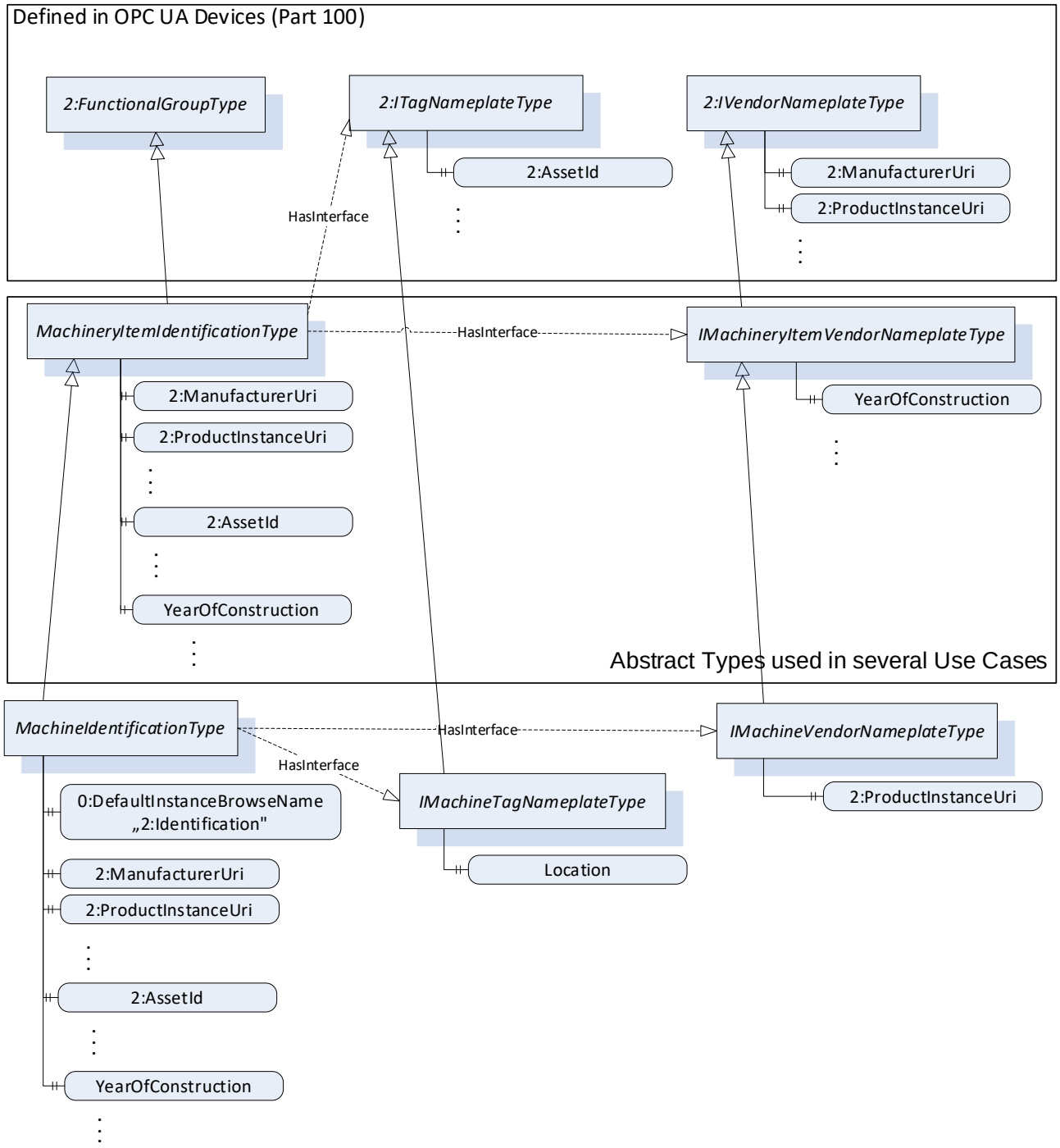

Figure 8 – Building Block for Identification and Nameplate

8.2 IMachineryItemVendorNameplateType

The *IMachineryItemVendorNameplateType* is a subtype of the *2:IVendorNameplateType* defined in OPC 10000-100. It refines the semantics of the *Properties* defined in *2:IVendorNameplateType*, makes some *Properties* mandatory and adds additional *Properties*. It is formally defined in Table 13.

Table 13 – IMachineryItemVendorNameplateType Definition

Attribute	Value				
BrowseName	IMachineryItemVendorNameplateType				
IsAbstract	True				
Description	Interface containing identification and nameplate information for a MachineryItem provided by the vendor				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 2:IVendorNameplateType defined in OPC 10000-100, i.e. inheriting the InstanceDeclarations of that Node.					
Properties of the 2:IVendorNameplateType					
0:HasProperty	Variable	2:Manufacturer	0:LocalizedText	0:PropertyType	M, RO
0:HasProperty	Variable	2:SerialNumber	0:String	0:PropertyType	M, RO
0:HasProperty	Variable	YearOfConstruction	0:UInt16	0:PropertyType	O, RO
0:HasProperty	Variable	MonthOfConstruction	0:Byte	0:PropertyType	O, RO
0:HasProperty	Variable	InitialOperationDate	0:DateTime	0:PropertyType	O, RO
Conformance Units					
Machinery Machine Identification					
Machinery Component Identification					

The mandatory *2:Manufacturer* provides a human-readable, localized name of the manufacturer. It is defined by the *2:IVendorNameplateType*. It is recommended to provide a language neutral *LocalizedText*. Clients shall not assume the uniqueness of the manufacturer based on this information, i.e. potentially several manufacturers use the same name. The value of this *Property* might change during the life-cycle of a *MachineryItem*, for example, when the name of the manufacturer changes due to an acquisition. The manufacturer might change this information, for example, within the next firmware update. Examples are “{“”,”TRUMPF”}” “{“”,”KUKA Deutschland GmbH”}”, “{“”, “ENGEL AUSTRIA GMBH”}”.

The mandatory *2:SerialNumber* is a string containing a unique production number of the manufacturer of the *MachineryItem*. It is defined by the *2:IVendorNameplateType*. The global uniqueness of the serial number is only given in the context of the manufacturer, and potentially the model. The value of this *Property* shall not change during the life-cycle of the *MachineryItem*. If a manufacturer internally does not manage serial numbers, as for example for special purpose machinery manufacturers, they could use for example the order number as serial number. Examples are: “A3231E001”, “643872”, “235223”.

The optional *YearOfConstruction* provides the year (Gregorian calendar) in which the manufacturing process of the *MachineryItem* has been completed. It shall be a four-digit number and never change during the life-cycle of a *MachineryItem*. For example: “2019”, “2020”.

The optional *MonthOfConstruction* provides the month in which the manufacturing process of the *MachineryItem* has been completed. It shall be a number between 1 and 12, representing the month from January to December. The *MonthOfConstruction* shall only be provided, if the *YearOfConstruction* is provided as well. For example, “1”, “2”, “3”.

The optional *InitialOperationDate* provides the date, when the *MachineryItem* was switched on the first time after it has left the manufacturer plant. The value of *InitialOperationDate* is not meant to provide any information about the state of warranty. If the date is not provided by the *MachineryItem*, the *InitialOperationDate* should not be added. The *InitialOperationDate* should be provided as UTC time. For example: “2020-01-29T18:59:59Z”, “2022-11-1712:00:00Z”.

How the *InitialOperationDate* is set is vendor-specific. This might be done by some manual configuration or can be done automatically. If the *InitialOperationDate* is set automatically, the vendor needs to ensure, that it uses a coordinated system time.

The optional *2:ProductInstanceUri* is a globally unique resource identifier provided by the manufacturer of the *MachineryItem*. It is defined by the *2:IVendorNameplateType*. It is intended to uniquely identify the *MachineryItem* and shall not change during the life-cycle of the *MachineryItem*. The length is restricted to 255 characters and it is the responsibility of the manufacturer that the *2:ProductInstanceUri* is globally unique. The recommended syntax of the *2:ProductInstanceUri* is: <ManufacturerUri>/<any string>. The manufacturer might choose the serial number of the *MachineryItem* as <any string>, if the serial number is unique within the manufacturer’s scope, or a combination of model and serial number, if the serial number is only unique within a model. Examples are: “http://www.trumpf.com/A3231E001”, “http://www.kuka.com/KR210R2700_EXTRA_C4_FLR/667659”, “http://www.engelglobal.com/Viper06/235223”.

The optional *2:ManufacturerUri* is a globally unique identifier of the manufacturer of the *MachineryItem*. It is defined by the *2:IVendorNameplateType*. It is intended to uniquely identify the manufacturer. It is the manufacturers responsibility to use the same identifier across its products. If the 2:*ManufacturerUri* is provided, it is recommended to be used as Prefix in the *2:ProductInstanceUri*. As *2:ManufacturerUri* is recommended to be used inside the *2:ProductInstanceUri*, it shall not change during the life-cycle of the *MachineryItem*, even if the manufacturer changes its name, e.g. due to an acquisition. Examples are: “http://www.trumpf.com”, “http://www.kuka.com”, “http://www.engelglobal.com”.

The optional 2:*Model* provides a human-readable, localized name of the model of the *MachineryItem*. It is defined by the 2:*IVendorNameplateInterfaceType*. If there is no specific model, this *Property* should not be provided. If the physical nameplate on the *MachineryItem* provides a model, the *Property* shall be provided. It is recommended to provide a language neutral *LocalizedText*. Examples are “{“”,“TruLaser 5030 (L76)”}”, “{“”,“VC 200/50”}”, “{“”,”KR210R2700EXTRAC4FLR”}”, “{“”,”Viper 6”}”.

The optional 2:*ProductCode* provides a machine-readable string of the model of the *MachineryItem*, that might include options like the hardware configuration of the model. This information might be provided by the ERP system of the vendor. For example, it can be used as order information. It is defined by the *2:IVendorNameplateType*. If no specific information is available, the *Property* should not be provided. The value of this *Property* shall not change during the life-cycle of the *MachineryItem*. Examples are “11182372”, “2377636”.

The optional *2:HardwareRevision* provides a string representation of the revision level of the hardware of a *MachineryItem*. Hardware is physical equipment, as opposed to programs, procedures, rules and associated documentation (see IEC 61499-1, [2]). The *Property* is defined by the *2:IVendorNameplateType*. Many *Machines* will not provide such information due to the modular and configurable nature of the *Machine*. The value of this *Property* might change during the life-cycle of a *MachineryItem*. Examples are: “01.33”, “A2”, “014/15120129-2018”.

The optional 2:*SoftwareRevision* provides a string representation of the overall software revision level of a *MachineryItem*. It is defined by the *2:IVendorNameplateType*. In most cases, *MachineryItems* consist of several software components. In that case, information about the software components might be provided as additional information in the *AddressS*pace, including individual revision information. The *2:SoftwareRevision* is either not provided or provides an overall software revision level. The value of this *Property* might change during the life-cycle of a *MachineryItem*. Examples are: “PLL01 1.10.0.3” “V05.01.01.15”, “3.1 R1293”, “70.0.1”, “4.60.03”.

The optional *Properties 2:DeviceRevision*, *2:RevisionCounter* and *2:DeviceManual* defined by the *2:IVendorNameplateType* are not further defined in this Interface.

The optional *2:DeviceClass*, defined by the *2:IVendorNameplateType*, should only be used, when a companion specification defines concrete values for specific *MachineryItem* classes. This specification does not define any values for this *Property*. Examples are: “Injection Moulding Machine”, “Drilling Machine”.

The *InstanceDeclarations* of the *IMachineryItemVendorNameplateType* have additional *Attribute* values defined in Table 14.

Table 14 – IMachineryItemVendorNameplateType Attribute Values for Child Nodes

SourceBrowsePath	Value	Description
2:Manufacturer	-	A human-readable, localized name of the manufacturer of the MachineryItem.
2:SerialNumber	-	A string containing a unique production number of the manufacturer of the MachineryItem. The global uniqueness of the serial number is only given in the context of the manufacturer, and potentially the model. The value shall not change during the life-cycle of the MachineryItem.
YearOfConstruction	-	The year (Gregorian calendar) in which the manufacturing process of the MachineryItem has been completed. It shall be a four-digit number and never change during the life-cycle of a MachineryItem.
MonthOfConstruction	-	The month in which the manufacturing process of the MachineryItem has been completed. It shall be a number between 1 and 12, representing the month from January to December.
InitialOperationDate	-	The date, when the MachineryItem was switched on the first time after it has left the manufacturer plant.

8.3 MachineryItemIdentificationType ObjectType Definition

The *MachineryItemIdentificationType* is an abstract *ObjectType* and cannot be used directly. It provides identification and other identification information of a *MachineryItem* and is formally defined in Table 21.

Table 15 – MachineryItemIdentificationType Definition

Attribute	**Value**				
BrowseName	MachineryItemIdentificationType				
IsAbstract	True				
Description	Contains information about the identification and nameplate of a MachineryItem				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 2:FunctionalGroupType defined in OPC 10000-100, i.e. inheriting the InstanceDeclarations of that Node.					
0:HasInterface	ObjectType	IMachineryItemVendorNameplateType			
0:HasInterface	ObjectType	2:ITagNameplateType			
0:HasProperty	Variable	0:DefaultInstanceBrowseName	0:QualifiedName	0:PropertyType	
Applied from IMachineryItemVendorNameplateType					
0:HasProperty	Variable	2:ProductInstanceUri	0:String	0:PropertyType	O, RO
0:HasProperty	Variable	2:Manufacturer	0:LocalizedText	0:PropertyType	M, RO
0:HasProperty	Variable	2:ManufacturerUri	0:String	0:PropertyType	O, RO
0:HasProperty	Variable	2:Model	0:LocalizedText	0:PropertyType	O, RO
0:HasProperty	Variable	2:ProductCode	0:String	0:PropertyType	O, RO
0:HasProperty	Variable	2:HardwareRevision	0:String	0:PropertyType	O, RO
0:HasProperty	Variable	2:SoftwareRevision	0:String	0:PropertyType	O, RO
0:HasProperty	Variable	2:DeviceClass	0:String	0:PropertyType	O, RO
0:HasProperty	Variable	2:SerialNumber	0:String	0:PropertyType	M, RO
0:HasProperty	Variable	YearOfConstruction	0:UInt16	0:PropertyType	O, RO
0:HasProperty	Variable	MonthOfConstruction	0:Byte	0:PropertyType	O, RO
0:HasProperty	Variable	InitialOperationDate	0:DateTime	0:PropertyType	O, RO
Applied from 2:ITagNameplateType					
0:HasProperty	Variable	2:AssetId	0:String	0:PropertyType	O, RW
0:HasProperty	Variable	2:ComponentName	0:LocalizedText	0:PropertyType	O, RW
Conformance Units					
Machinery Machine Identification					
Machinery Component Identification					

The *Properties 2:ProductInstanceUri*, *2:Manufacturer*, *2:ManufacturerUri*, *2:Model*, *2:ProductCode*, *2:HardwareRevision*, *2:SoftwareRevision*, *2:DeviceClass*, *2:SerialNumber*, *YearOfConstruction*, and *InitialOperationDate* are defined by the *IMachineryItemVendorNameplateType* and shall be used as defined by the Interface.

In some subtypes it is not recommended to use the *Properties 2:DeviceRevision*, *2:RevisionCounter* and *2:DeviceManual* defined by the *2:IVendorNameplateType* and inherited by the *IMachineVendorNameplateType.* Therefore, those optional *Properties* are not applied on the *ObjectType*. Subtypes of this *ObjectType* might add those *Properties*.

The *Properties 2:AssetId* and *2:ComponentName* are defined by the *2:ITagNameplateType* and shall be used as defined by the Interface.

The *InstanceDeclarations* of the *MachineryItemIdentificationType* have additional *Attribute* values defined in Table 22.

Table 16 – MachineryItemIdentificationType Attribute Values for Child Nodes

SourceBrowsePath	Value	Description
0:DefaultInstanceBrowseName	2:Identification	The default BrowseName for instances of the type.
2:ProductInstanceUri	-	A globally unique resource identifier provided by the manufacturer of the MachineryItem.
2:Manufacturer	-	A human-readable, localized name of the manufacturer of the MachineryItem.
2:ManufacturerUri	-	A globally unique identifier of the manufacturer of the MachineryItem.
2:Model	-	A human-readable, localized name of the model of the MachineryItem.
2:ProductCode	-	A machine-readable string of the model of the MachineryItem, that might include options like the hardware configuration of the model. This information might be provided by the ERP system of the vendor. For example, it can be used as order information.
2:HardwareRevision	-	A string representation of the revision level of the hardware of a MachineryItem. Hardware is physical equipment, as opposed to programs, procedures, rules and associated documentation. Many machines will not provide such information due to the modular and configurable nature of the machine.
2:SoftwareRevision	-	A string representation of the revision level of a MachineryItem. In most cases, MachineryItems consist of several software components. In that case, information about the software components might be provided as additional information in the address space, including individual revision information. In that case, this property is either not provided or provides an overall software revision level. The value might change during the life-cycle of a MachineryItem.
2:DeviceClass	-	Indicates in which domain or for what purpose the MachineryItem is used.
2:SerialNumber	-	A string containing a unique production number of the manufacturer of the MachineryItem. The global uniqueness of the serial number is only given in the context of the manufacturer, and potentially the model. The value shall not change during the life-cycle of the MachineryItem.
YearOfConstruction	-	The year (Gregorian calendar) in which the manufacturing process of the MachineryItem has been completed. It shall be a four-digit number and never change during the life-cycle of a MachineryItem.
MonthOfConstruction	-	The month in which the manufacturing process of the MachineryItem has been completed. It shall be a number between 1 and 12, representing the month from January to December.
InitialOperationDate	-	The date, when the MachineryItem was switched on the first time after it has left the manufacturer plant.
2:AssetId	“”	To be used by end users to store a unique identification in the context of their overall application. Servers shall support at least 40 Unicode characters for the clients writing this value, this means clients can expect to be able to write strings with a length of 40 Unicode characters into that field.
2:ComponentName	“”	To be used by end users to store a human-readable localized text for the MachineryItem. The minimum number of locales supported for this property shall be two. Servers shall support at least 40 Unicode characters for the clients writing the text part of each locale, this means clients can expect to be able to write texts with a length of 40 Unicode characters into that field.

8.4 IMachineVendorNameplateType

The *IMachineVendorNameplateType* is a subtype of the *IMachineryItemVendorNameplateType*. It makes one *Property* mandatory. It is formally defined in Table 17.

Table 17 – IMachineVendorNameplateType Definition

Attribute	Value				
BrowseName	IMachineVendorNameplateType				
IsAbstract	True				
Description	Interface containing identification and nameplate information for a machine provided by the machine vendor				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the IMachineryItemVendorNameplateType defined in 8.2, i.e. inheriting the InstanceDeclarations of that Node.					
Properties of the 2:IVendorNameplateType					
0:HasProperty	Variable	2:ProductInstanceUri	0:String	0:PropertyType	M, RO
Conformance Units					
Machinery Machine Identification					

The *Properties* defined by the *IMachineryItemVendorNameplateType* shall be used as defined by that Interface.

It is not recommended to use the optional *Properties 2:DeviceRevision*, *2:RevisionCounter* and *2:DeviceManual* defined by the *2:IVendorNameplateType.*

The *InstanceDeclarations* of the *IMachineVendorNameplateType* have additional *Attribute* values defined in Table 18.

Table 18 – IMachineVendorNameplateType Attribute Values for Child Nodes

SourceBrowsePath	Value	Description
2:ProductInstanceUri	-	A globally unique resource identifier provided by the manufacturer of the machine

8.5 IMachineTagNameplateType

The *IMachineTagNameplateType* is a subtype of the *2:ITagNameplateType* defined in OPC 10000-100. It refines the semantics of the *Properties* defined in *2:ITagNameplateType*, and adds an additional *Property*. It is formally defined in Table 19.

Table 19 – IMachineTagNameplateType Definition

Attribute	Value				
BrowseName	IMachineTagNameplateType				
IsAbstract	True				
Description	Interface containing information of the identification of a machine set by the customer				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 2:ITagNameplateType defined in OPC 10000-100, i.e. inheriting the InstanceDeclarations of that Node.					
0:HasProperty	Variable	Location	0:String	0:PropertyType	O, RW
Conformance Units					
Machinery Machine Identification					

The optional 2:*AssetId* is a writable string. It is defined by the *2:ITagNameplateType*. The default value shall be an empty string. The *Property* is intended to be used by end users to store a unique identification in the context of their overall application. Servers shall support at least 40 Unicode characters for the clients writing this value, this means clients can expect to be able to write strings with a length of 40 Unicode characters into that field.

The optional *2:ComponentName* is a writable localized text. It is defined by the *2:ITagNameplateType*. The default value shall be an empty string for locale and text. The *Property* is intended to be used by end users to store a human-readable localized text for the *Machine*. The minimum number of locales supported for this *Property* shall be two. Servers shall support at least 40 Unicode characters for the clients writing the text part of each locale, this means clients can expect to be able to write texts with a length of 40 Unicode characters into that field.

The optional *Location* is a writable string. The Property is intended to be used by end users to store the location of the *Machine* in a scheme specific to the end user. Servers shall support at least 60 Unicode characters for the clients writing this value, this means clients can expect to be able to write strings with a length of 60 Unicode characters into that field. Examples are “Munich/A2/217”, “Area 51”.

The *InstanceDeclarations* of the *IMachineTagNameplateType* have additional *Attribute* values defined in Table 20.

Table 20 – IMachineTagNameplateType Attribute Values for Child Nodes

SourceBrowsePath	Value	Description
Location	-	To be used by end users to store the location of the machine in a scheme specific to the end user Servers shall support at least 60 Unicode characters for the clients writing this value, this means clients can expect to be able to write strings with a length of 60 Unicode characters into that field.

8.6 MachineIdentificationType ObjectType Definition

The *MachineIdentificationType* provides a globally unique identification of a *Machine* and other identification information of a *Machine* and is formally defined in Table 21.

Table 21 – MachineIdentificationType Definition

Attribute	Value				
BrowseName	MachineIdentificationType				
IsAbstract	False				
Description	Contains information about the identification and nameplate of a machine				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the MachineryItemIdentificationType defined in 8.3, i.e. inheriting the InstanceDeclarations of that Node.					
0:HasProperty	Variable	0:DefaultInstanceBrowseName	0:QualifiedName	0:PropertyType	-
0:HasInterface	ObjectType	IMachineVendorNameplateType			
0:HasInterface	ObjectType	IMachineTagNameplateType			
Applied from IMachineVendorNameplateType					
0:HasProperty	Variable	2:ProductInstanceUri	0:String	0:PropertyType	M, RO
Applied from IMachineTagNameplateType					
0:HasProperty	Variable	Location	0:String	0:PropertyType	O, RW
Conformance Units					
Machinery Machine Identification					

The *Properties 2:ProductInstanceUri*, *2:Manufacturer*, *2:ManufacturerUri*, *2:Model*, *2:ProductCode*, *2:HardwareRevision*, *2:SoftwareRevision*, *2:DeviceClass*, *2:SerialNumber*, *YearOfConstruction* and *InitialOperationDate* are defined by the *IMachineVendorNameplateType* and already inherited from the *MachineryItemIdentificationType* and shall be used as defined by both. The *2:ProductInstanceUri* is mandatory.

It is not recommended to use the optional *Properties 2:DeviceRevision*, *2:RevisionCounter* and *2:DeviceManual* defined by the *2:IVendorNameplateType* and inherited by the *IMachineVendorNameplateType*. Therefore, those optional *Properties* are not applied on the *ObjectType*.

The *Properties 2:AssetId*, *2:ComponentName*, and *Location* are defined by the *IMachineTagNameplateType* and shall be used as defined by the Interface. *2:AssetId* and *2:ComponentName* are also inherited from the MachineryItemIdentificationType.

The *InstanceDeclarations* of the *MachineIdentificationType* have additional *Attribute* values defined in Table 22.

Table 22 – MachineIdentificationType Attribute Values for Child Nodes

SourceBrowsePath	Value	Description
0:DefaultInstanceBrowseName	2:Identification	The default BrowseName for instances of the type.
2:ProductInstanceUri	-	A globally unique resource identifier provided by the manufacturer of the machine
Location	-	To be used by end users to store the location of the machine in a scheme specific to the end user. Servers shall support at least 60 Unicode characters for the clients writing this value, this means clients can expect to be able to write strings with a length of 60 Unicode characters into that field.

9 Finding all Machines in a Server

9.1 Overview

An OPC UA Server may contain many *Nodes*, organized in vendor-specific ways. The OPC UA specification already defines entry points to start browsing instances or types. However, finding specific *Nodes* might be challenging since they may be managed somewhere inside the hierarchies of *Nodes* of the OPC UA Server.

This building block provides the capability to easily find all *Machines* managed in a *Server*. Figure 9 gives an overview. There is a well-defined *Object* in the *AddressSpace* as entry point to browse to *Objects* representing a *Machine*.

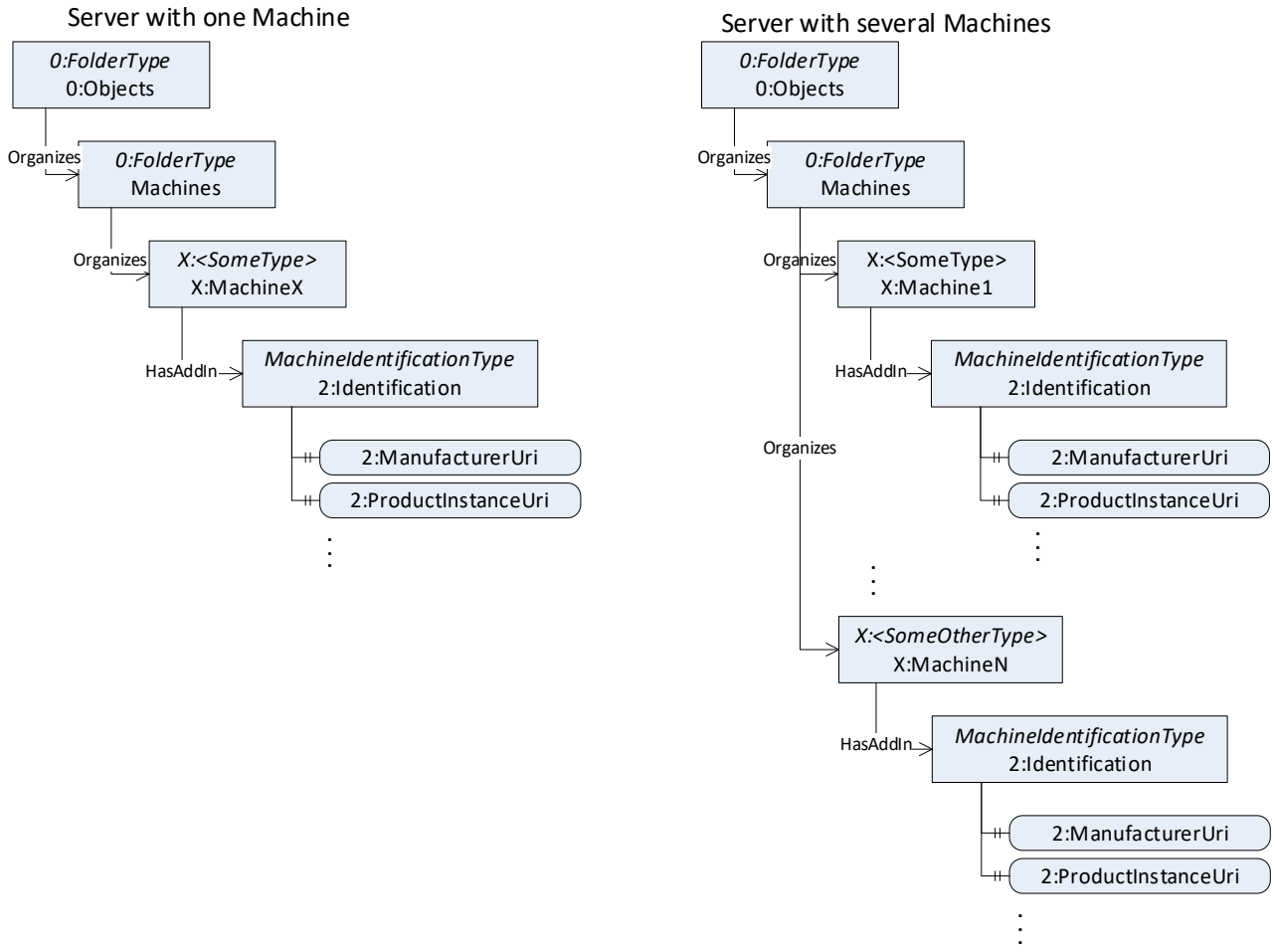

Figure 9 – Building Block for Finding all Machines in Server

In many cases, *Servers* will only manage one *Machine*. For example, if the *Server* runs on the PLC of a particular *Machine*. However, *Servers* can also manage several *Machines*, for example, in a cell or production line, or a robot system consisting of a controller and a robot, or when aggregating *Machines* of a factory floor, a factory, or company-wide.

9.2 Machines Object Definition

The Machines *Object* is a standardized entry point to access all *Machines* managed in the *Server* and formally defined in Table 23. All *Objects* representing *Machines*, that are managed in the *Server*, shall be referenced directly from this *Object* with a *Reference* of *ReferenceType Organizes* or a subtype of *Organizes*.

Table 23 – Machines Definition

Attribute	**Value**			
BrowseName	Machines			
Description	This object is the entry point to machines managed in the server. All machines are directly referenced by this object.			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**
OrganizedBy by the 0:Objects defined in OPC 10000-5				
0:HasTypeDefinition	ObjectType	0:FolderType	Defined in OPC 10000-5	
Conformance Units				
Machinery Find Machines				

In order to identify the referenced *Objects* as representations of *Machines*, each of those *Objects* shall provide the *MachineIdentificationType AddIn* using the *0:DefaultInstanceBrowseName* as a direct sub-component of the *Object* (referenced with a Reference of *ReferenceType 0:HasAddIn* or a subtype of it).

The *Machines Object* shall be referenced from the *0:Objects Object* defined in OPC 10000-5 with an *Organizes Reference*.

Since later versions of this specification might change the parent of this *Object*, *Clients* aware of this standardized *Object* shall not access it via its parent but directly via its standardized *NodeId*.

10 Component Identification and Nameplate

10.1 Overview

This building block provides the capabilities to identify components of a *Machine* and have access to vendor-defined information about the components and manage user-specific information for the identification of the component.

Figure 10 gives an overview. The *AddIn MachineryComponentIdentificationType* with the default name "2:Identification" (as defined in OPC 10000-100) is derived from the *MachineryItemIdentificationType* and thus indirectly from the *2:FunctionalGroupType*. It adds *Properties* and refines the semantics of the inherited *Properties*.

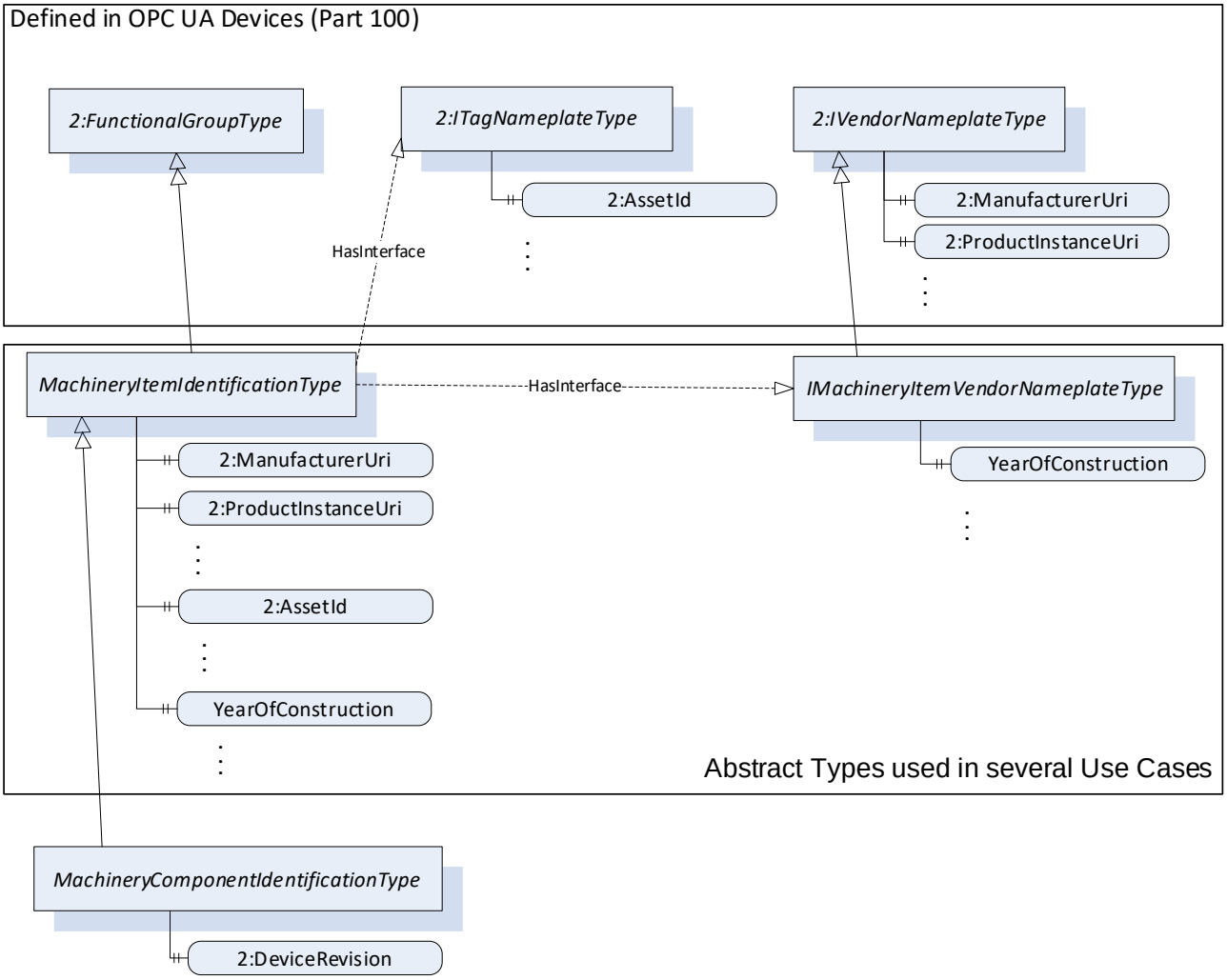

Figure 10 – Building Block for Component Identification and Nameplate

10.2 MachineryComponentIdentificationType ObjectType Definition

The *MachineryComponentIdentificationType* provides identification information about a component, like manufacturer or serial number, and allows setting user-specific information like *2:ComponentName* and is formally defined in Table 21.

Table 24 – MachineryComponentIdentificationType Definition

Attribute	**Value**				
BrowseName	MachineryComponentIdentificationType				
IsAbstract	False				
Description	Contains information about the identification and nameplate of a component				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the MachineryItemIdentificationType defined in 8.3, i.e. inheriting the InstanceDeclarations of that Node.					
0:HasProperty	Variable	0:DefaultInstanceBrowseName	0:QualifiedName	0:PropertyType	-
0:HasProperty	Variable	2:DeviceRevision	0:String	0:PropertyType	O, RO
Conformance Units					
Machinery Component Identification					

The optional 2:*DeviceRevision* is defined by the 2:IVendorNameplateType, but not included in the *MachineryItemIdentificationType*. Therefore, it is added to the *MachineryComponentIdentificationType*. It provides the overall revision level of the component. Often, it is increased when either the *SoftwareRevision* and / or the *2:HardwareRevision* of the component is increased. As an example, this *Property* can be used in ERP systems together with the *2:ProductCode Property.* Examples are: “PLL01 1.10.0.3” “V05.01.01.15”, “3.1 R1293”, “70.0.1”, “4.60.03”.

It is not recommended to use the optional *Properties 2:RevisionCounter* and *2:DeviceManual* defined by the *2:IVendorNameplateType* and inherited by the *IMachineVendorNameplateType*.

A *Machine* vendor should not generate a *2:ProductInstanceUri* for components of its *Machine*, but take the one provided by the component manufacturer. If the component manufacturer does not provide such a *2:ProductInstanceUri*, the *Property* shall be omitted. *Clients* can use the *2:ProductInstanceUri* of the *Machine* in combination with the *NodeId* of the component to generate a globally unique identification, which is only valid in the context of the *Machine*. Examples are: http://www.componentvendor.de/A3231E001.

A *Machine* vendor should not change the *InitialOperationDate* for components of its *Machine*, but take the one provided by the component manufacturer.

Machine vendors might use the *2:AssetId* to provide their internal machine-readable identification of the component. In that case, the *Property* might be provided as read-only.

Machine vendors shall not use the *2:ComponentName Property* to manage its identification of the component. Instead of, the *BrowseName* and *DisplayName Attributes* of the *Object* representing the component shall be used for that purpose.

The *InstanceDeclarations* of the *MachineryComponentIdentificationType* have additional *Attribute* values defined in Table 25.

Table 25 – MachineryComponentIdentificationType Attribute Values for Child Nodes

SourceBrowsePath	**Value**	**Description**
0:DefaultInstanceBrowseName	2:Identification	The default BrowseName for instances of the type.
2:DeviceRevision	-	A string representation of the overall revision level of the component. Often, it is increased when either the SoftwareRevision and / or the HardwareRevision of the component is increased. As an example, it can be used in ERP systems together with the ProductCode.

11 Finding all identifiable Components of a Machine

11.1 Overview

Machines may organize their components in many different ways. Therefore, it might be challenging to find all components that are part of a *Machine*.

This building block provides the capability to easily find all identifiable components (providing the Identification Object) of a *Machine*. Figure 11 gives an overview. Each *Machine* providing this building block provides the Components *Object*, which directly references all identifiable components of a *Machine*.

Note: That does not preclude that a *Machine* is organizing its components in various other hierarchies as well.

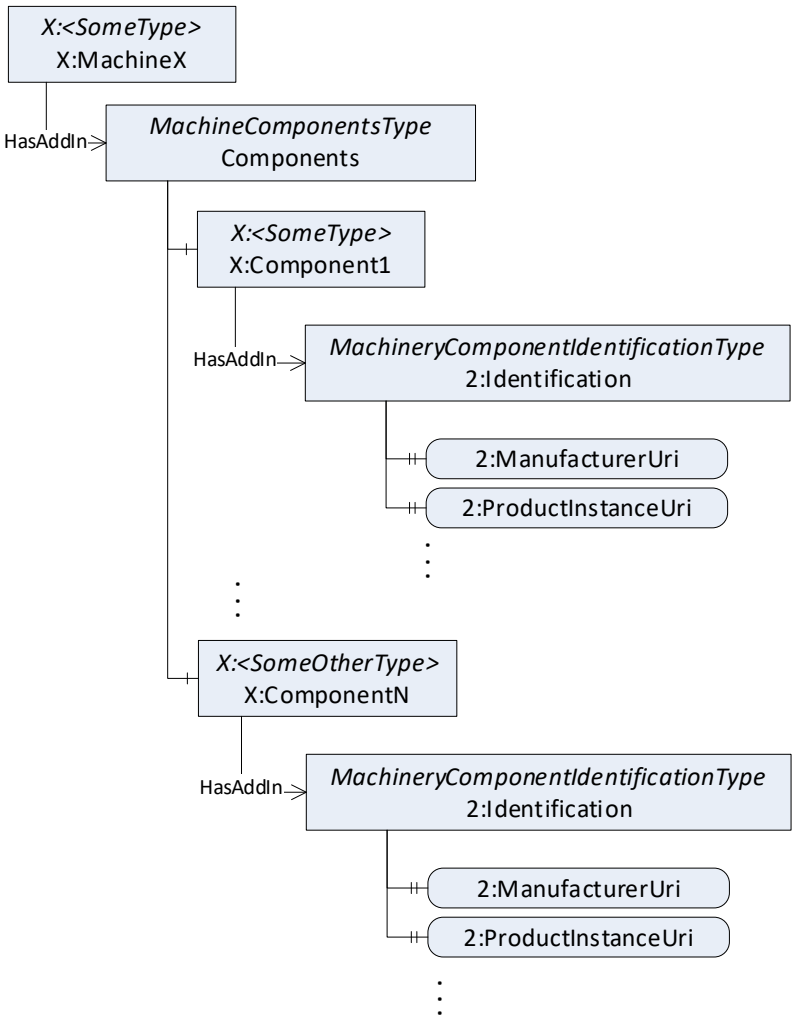

Figure 11 – Building Block for Finding all Identifiable Components of a Machine

As a special case, a *Machine* might contain other *Machines* as components. In that case, the contained Machine is referenced from the Components *Object* like any other component, as shown in Figure 12. In addition, the contained *Machine* is also referenced from the *Machines Object* directly, as also shown in the Figure.

Note: Consider the example of a *Machine* MachineA containing a *Machine* MachineB. Components of the contained MachineB are typically only considered to be components of MachineB, that is, they are typically not directly referenced from the Components *Object* of the containing MachineA.

Note: It is expected that domain-specific companion specifications using this specification will define what assets of the domain in what usage are considered to be *Machines* or components of *Machines*.

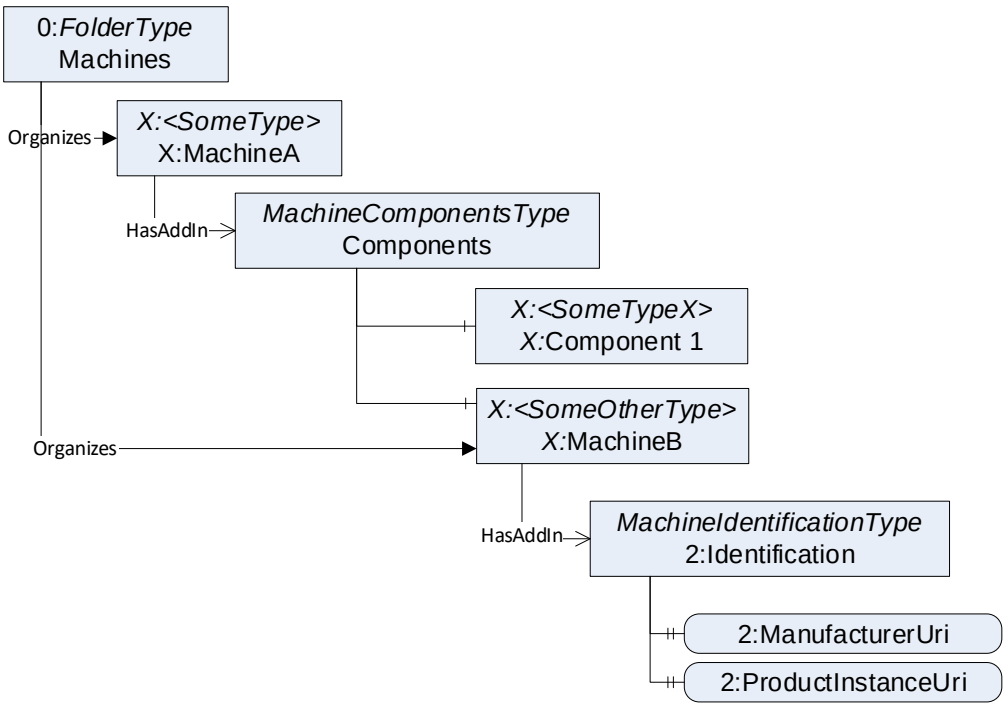

Figure 12 – Example of a Machine containing a Machine

11.2 MachineComponentsType ObjectType Definition

The *MachineComponentsType* provides *HasComponent References* to all identifiable components of a *Machine* and is formally defined in Table 26.

Table 26 – MachineComponentsType Definition

Attribute	**Value**				
BrowseName	MachineComponentsType				
IsAbstract	False				
Description	Contains all identifiable components of a machine				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:BaseObjectType defined in OPC 10000-5, i.e. inheriting the InstanceDeclarations of that Node.					
0:HasProperty	Variable	0:DefaultInstanceBrowseName	0:QualifiedName	0:PropertyType	-
0:HasComponent	Object	<Component>		0:BaseObjectType	OP
Conformance Units					
Machinery Find Components of Machines					

The <*Component*> *Object* has the *ModellingRule OptionalPlaceholder* and represents the identifiable components of a *Machine*. As any *ObjectType* can be used for components, it is of *BaseObjectType*. As all identifiable components shall contain the Identification *Object*, <Component> references to that *AddIn* as defined in Table 27. As a *Machine* can contain other *Machines*, it does not reference to the *MachineryComponentIdentificationType*, but the *MachineryItemIdentificationType*, which is the abstract supertype of *MachineryComponentIdentificationType* and *MachineIdentificationType*.

Table 27 – MachineComponentsType Additional Subcomponents

SourceBrowsePath	**References**	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Others**
<Component>	0:HasAddIn	Object	2:Identification		MachineryItemIdentificationType	M

The *InstanceDeclarations* of the *MachineComponentsType* have additional *Attribute* values defined in Table 28.

Table 28 – MachineComponentsType Attribute Values for Child Nodes

SourceBrowsePath	Value	Description
0:DefaultInstanceBrowseName	Components	The default BrowseName for instances of the type.
<Component>	-	Represents the identifiable components of a machine.

12 MachineryItemState

12.1 Overview

This building block provides information about the state of a *MachineryItem*. It defines some common top-level states. Companion specifications or vendors might extend those states with substates, but cannot add additional top-level states. In order to have a modelling mechanism that allows the creation of substate, but also to define causes and effects on specific transitions, the concept of a *FiniteStateMachine* defined in OPC 10000-16 is used. Since the intention of this base specification is not to restrict the *Transitions* between the *States*, the *StateMachine* defines *Transitions* between all *States*. Instances might restrict the usage of the defined *States* and *Transitions*. That might be used by domain-specific companion specifications to restrict the usage of the *StateMachine*. An overview of the *StateMachine* is given in Figure 13. The *States* are further described in 12.2.

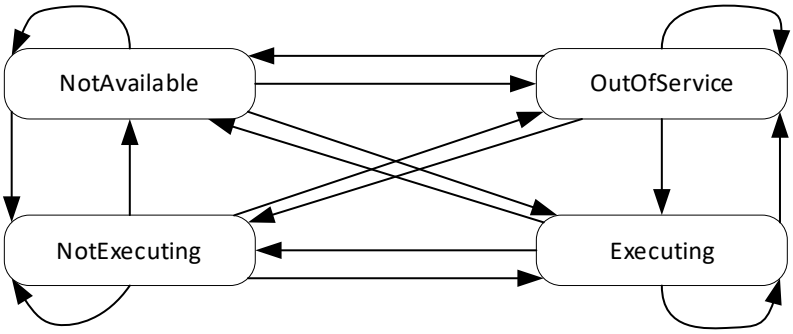

Figure 13 – MachineryItemState StateMachine

The overview of the building block is shown in Figure 14. The *MachineryItemState_StateMachineType* is a subtype of the *FiniteStateMachineType* defining the *States* and *Transitions* of that *StateMachine*. The usage of the building block is also shown in that figure. An instance of the *MachineryItemState_StateMachineType* is added to the *Object* MyMachine representing a *Machine* using the *AddIn* concept. The *MachineryItemState_StateMachineType* is defined in 12.2.

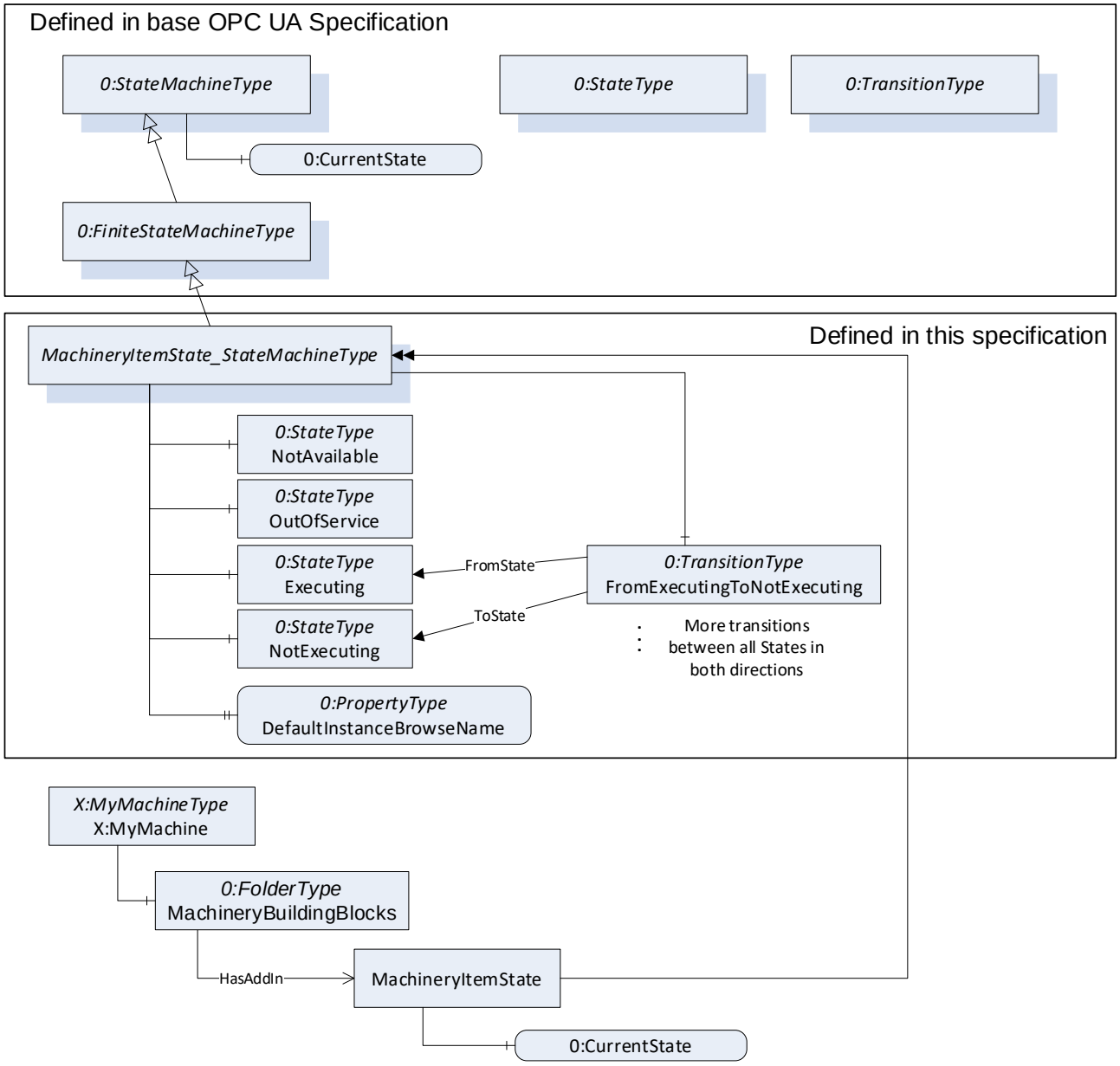

Figure 14 – Building Block for MachineryItemState

In Figure 15, a more complex example is shown, where the *Machine* also contains a component having the *MachineryItemState*.

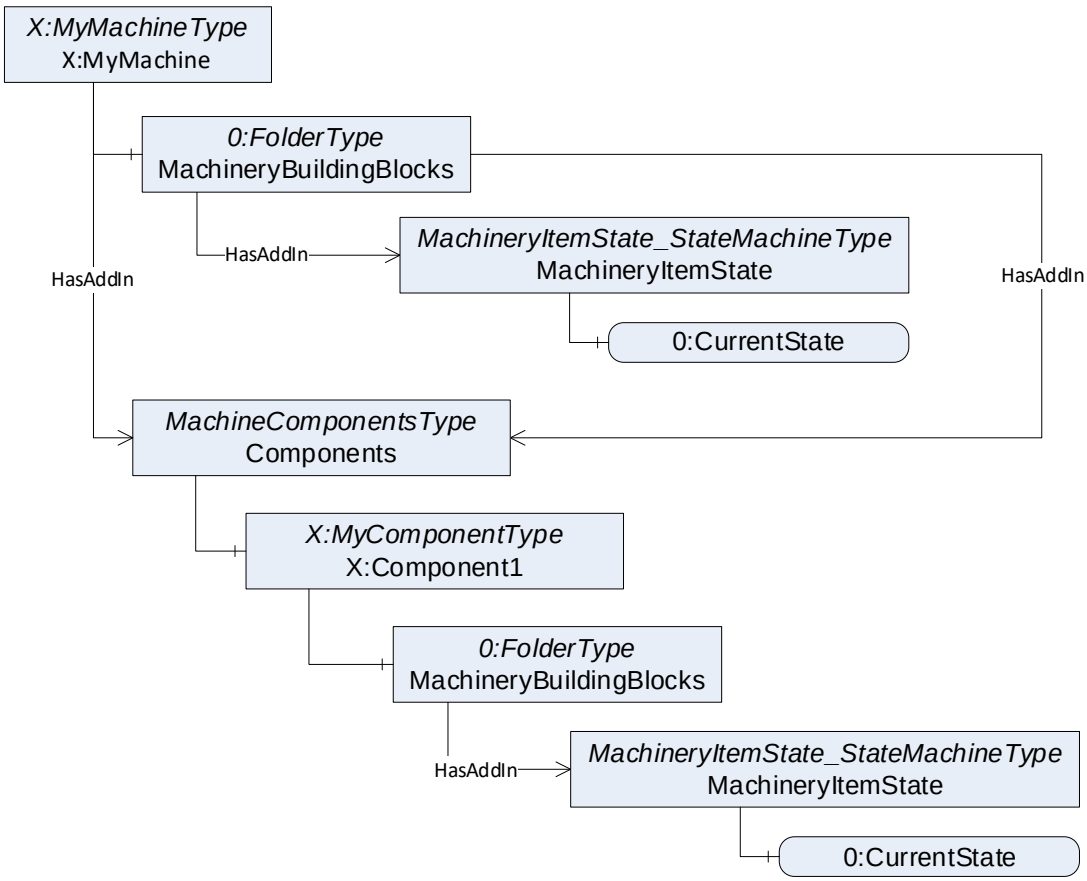

Figure 15 – Example of using the MachineryItem State

12.2 MachineryItemState_StateMachineType

The *MachineryItemState_StateMachineType* is a subtype of *FiniteStateMachineType* defined in OPC 10000-16. The semantic of this *AddIn* is to provide the state of a *MachineryItem*. It is formally defined in Table 29.

Table 29 – MachineryItemState_StateMachineType Definition

Attribute	Value				
BrowseName	MachineryItemState_StateMachineType				
IsAbstract	False				
Description	State machine representing the state of a machinery item				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:FiniteStateMachineType defined in OPC 10000-16, i.e. inheriting the InstanceDeclarations of that Node.					
0:HasProperty	Variable	0:DefaultInstanceBrowseName	0:QualifiedName	0:PropertyType	-
0:HasComponent	Object	NotAvailable	-	0:StateType	-
0:HasComponent	Object	OutOfService	-	0:StateType	-
0:HasComponent	Object	Executing	-	0:StateType	-
0:HasComponent	Object	NotExecuting	-	0:StateType	-
0:HasComponent	Object	FromNotAvailableToOutOfService	-	0:TransitionType	-
0:HasComponent	Object	FromNotAvailableToNotExecuting	-	0:TransitionType	-
0:HasComponent	Object	FromNotAvailableToExecuting	-	0:TransitionType	-
0:HasComponent	Object	FromNotAvailableToNotAvailable	-	0:TransitionType	-
0:HasComponent	Object	FromOutOfServiceToNotAvailable	-	0:TransitionType	-
0:HasComponent	Object	FromOutOfServiceToNotExecuting	-	0:TransitionType	-
0:HasComponent	Object	FromOutOfServiceToExecuting	-	0:TransitionType	-
0:HasComponent	Object	FromOutOfServiceToOutOfService	-	0:TransitionType	-
0:HasComponent	Object	FromNotExecutingToNotAvailable	-	0:TransitionType	-
0:HasComponent	Object	FromNotExecutingToOutOfService	-	0:TransitionType	-
0:HasComponent	Object	FromNotExecutingToExecuting	-	0:TransitionType	-
0:HasComponent	Object	FromNotExecutingToNotExecuting	-	0:TransitionType	-
0:HasComponent	Object	FromExecutingToNotAvailable	-	0:TransitionType	-
0:HasComponent	Object	FromExecutingToOutOfService	-	0:TransitionType	-
0:HasComponent	Object	FromExecutingToNotExecuting	-	0:TransitionType	-
0:HasComponent	Object	FromExecutingToExecuting	-	0:TransitionType	-
Conformance Units					
Machinery MachineryItem State					

The *States* and *Transitions* define a *StateMachine* as shown in Figure 13. It does not define an initial *State*, i.e., the initial *State* is vendor-specific.

Note: None of the *States* or *Transitions* have a *ModellingRule*, i.e., they are only provided in the *TypeDefinition*, not on the instance. The *CurrentState Variable* (inherited from *StateMachineType*) contains the information of the current *State* of the instance.

The *NotAvailable State* represents that the *MachineryItem* is not available and does not perform any activity. Examples for this *State* are that the *Machine* is switched off or is in energy saving mode.

Note: Activity can be part of the production, preparation or maintenance process.

Note: The *NotAvailable State* should only be used, when the *State* of the *Machine* is known. If the OPC UA *Server* is for example deployed outside the *Machine* and just lost the connection to the *Machine*, it should rather use a Bad *StatusCode* for the *CurrentState* than the *NotAvailable State*.

Note: Depending on the deployment of the OPC UA *Server*, the *NotAvailable State* might never be provided by the OPC UA *Server*, since the *Server* might not be available when the *Machine* is not available, e.g., if the OPC UA Server is deployed on a PLC of the *Machine*.

The *OutOfService State* represents that the *MachineryItem* is not functional and does not perform any activity. Examples for this *State* are that the *Machine* is in an error or blocked.

The *NotExecuting State* represents that the *MachineryItem* is available & functional and does not perform any activity. It waits for an action from outside to start or restart an activity. Examples for this *State* are that the *Machine* is waiting for a new order to be produced, a piece put into the *Machine* or an automatic or manual activation to be executed.

The *Executing State* represents that the *Machine* is available & functional and is actively performing an activity (pursues a purpose). Examples for this *State* are that the *Machine* is producing, transporting or processing something or executing a maintenance process.

The meaning of the *Transitions* is defined in the *Description* of Table 30.

The *InstanceDeclarations* of the *MachineryItemState_StateMachineType* have additional *Attribute* values defined in Table 30.

Table 30 – MachineryItemState_StateMachineType Attribute Values for Child Nodes

SourceBrowsePath	Value	Description
0:DefaultInstanceBrowseName	MachineryItemState	The default BrowseName for instances of the type
NotAvailable	-	The machine is not available and does not perform any activity (e.g., switched off, in energy saving mode)
OutOfService	-	The machine is not functional and does not perform any activity (e.g., error, blocked)
NotExecuting	-	The machine is available & functional and does not perform any activity. It waits for an action from outside to start or restart an activity
Executing	-	The machine is available & functional and is actively performing an activity (pursues a purpose)
FromNotAvailableToOutOfService	-	Transition from state NotAvailable to state OutOfService
FromNotAvailableToNotExecuting	-	Transition from state NotAvailable to state NotExecuting
FromNotAvailableToExecuting	-	Transition from state NotAvailable to state Executing
FromNotAvailableToNotAvailable	-	Transition from state NotAvailable to state NotAvailable
FromOutOfServiceToNotAvailable	-	Transition from state OutOfService to state NotAvailable
FromOutOfServiceToNotExecuting	-	Transition from state OutOfService to state NotExecuting
FromOutOfServiceToExecuting	-	Transition from state OutOfService to state Executing
FromOutOfServiceToOutOfService	-	Transition from state OutOfService to state OutOfService
FromNotExecutingToNotAvailable	-	Transition from state NotExecuting to state NotAvailable
FromNotExecutingToOutOfService	-	Transition from state NotExecuting to state OutOfService
FromNotExecutingToExecuting	-	Transition from state NotExecuting to state Executing
FromNotExecutingToNotExecuting	-	Transition from state NotExecuting to state NotExecuting
FromExecutingToNotAvailable	-	Transition from state Executing to state NotAvailable
FromExecutingToOutOfService	-	Transition from state Executing to state OutOfService
FromExecutingToNotExecuting	-	Transition from state Executing to state NotExecuting
FromExecutingToExecuting	-	Transition from state Executing to state Executing

NotAvailable 0:StateNumber	0	-
OutOfService 0:StateNumber	1	-
NotExecuting 0:StateNumber	2	-
Executing 0:StateNumber	3	-
FromNotAvailableToOutOfService 0:TransitionNumber	0	-
FromNotAvailableToExecuting 0:TransitionNumber	1	-
FromNotAvailableToNotExecuting 0:TransitionNumber	2	-
FromOutOfServiceToNotAvailable 0:TransitionNumber	3	-
FromOutOfServiceToExecuting 0:TransitionNumber	4	-
FromOutOfServiceToNotExecuting 0:TransitionNumber	5	-
FromExecutingToNotAvailable 0:TransitionNumber	6	-
FromExecutingToOutOfService 0:TransitionNumber	7	-
FromExecutingToNotExecuting 0:TransitionNumber	8	-
FromNotExecutingToNotAvailable 0:TransitionNumber	9	-
FromNotExecutingToOutOfService 0:TransitionNumber	10	-
FromNotExecutingToExecuting 0:TransitionNumber	11	-
FromNotAvailableToNotAvailable 0:TransitionNumber	12	-
FromOutOfServiceToOutOfService 0:TransitionNumber	13	-
FromExecutingToExecuting 0:TransitionNumber	14	-
FromNotExecutingToNotExecuting 0:TransitionNumber	15	-

The components of the *MachineryItemState_StateMachineType* have additional *References* which are defined in Table 31.

Table 31 – MachineryItemState_StateMachineType Additional References

SourceBrowsePath	Reference Type	Is Forward	TargetBrowsePath
FromNotAvailableToOutOfService	0:FromState	True	NotAvailable
	0:ToState	True	OutOfService
FromNotAvailableToExecuting	0:FromState	True	NotAvailable
	0:ToState	True	Executing
FromNotAvailableToNotExecuting	0:FromState	True	NotAvailable
	0:ToState	True	NotExecuting
FromOutOfServiceToNotAvailable	0:FromState	True	OutOfService
	0:ToState	True	NotAvailable
FromOutOfServiceToExecuting	0:FromState	True	OutOfService
	0:ToState	True	Executing
FromOutOfServiceToNotExecuting	0:FromState	True	OutOfService
	0:ToState	True	NotExecuting
FromExecutingToNotAvailable	0:FromState	True	Executing
	0:ToState	True	NotAvailable
FromExecutingToOutOfService	0:FromState	True	Executing
	0:ToState	True	OutOfService
FromExecutingToNotExecuting	0:FromState	True	Executing
	0:ToState	True	NotExecuting
FromNotExecutingToNotAvailable	0:FromState	True	NotExecuting
	0:ToState	True	NotAvailable
FromNotExecutingToOutOfService	0:FromState	True	NotExecuting
	0:ToState	True	OutOfService
FromNotExecutingToExecuting	0:FromState	True	NotExecuting
	0:ToState	True	Executing
FromNotAvailableToNotAvailable	0:FromState	True	NotAvailable
	0:ToState	True	NotAvailable
FromOutOfServiceToOutOfService	0:FromState	True	OutOfService
	0:ToState	True	OutOfService
FromExecutingToExecuting	0:FromState	True	Executing
	0:ToState	True	Executing
FromNotExecutingToNotExecuting	0:FromState	True	NotExecuting
	0:ToState	True	NotExecuting

13 MachineryOperationMode

13.1 Overview

This building block provides information about the *MachineryOperationMode* of a *MachineryItem*. It defines some common top-level modes. Companion specifications or vendors might extend those modes with submodes, but cannot add additional top-level modes. In order to have a modelling mechanism that allows the creation of submodes, but also to define causes and effects on specific transitions, the concept of a *FiniteStateMachine* defined in OPC 10000-16 is used. Since the intention of this base specification is not to restrict the *Transitions* between the *States*, the *StateMachine* defines *Transitions* between all *States*. Instances might restrict the usage of the defined *States* and *Transitions*. That might be used by domain-specific companion specifications to restrict the usage of the *StateMachine*. An overview of the *StateMachine* is given in Figure 16. The *States* are further described in 13.2.

Note that the *MachineryOperationMode* may not be known by the *MachineryItem* itself. In this case, the information needs to be provided by an external source like an MES system or the operator.

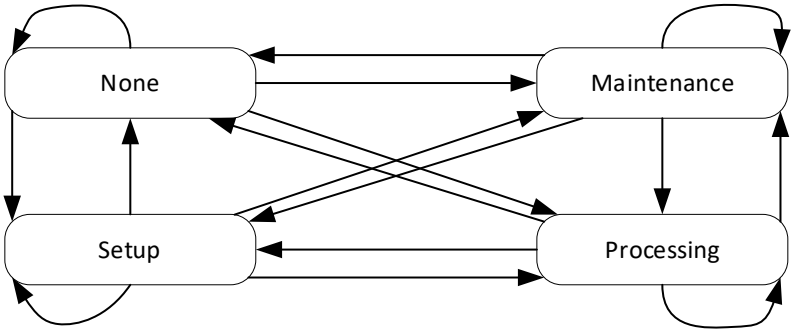

Figure 16 – MachineryOperationMode StateMachine

The overview of the building block is shown in Figure 17. The *MachineryOperationModeStateMachineType* is a subtype of the *FiniteStateMachineType* defining the *States* and *Transitions* of that *StateMachine*. The usage of the building block is also shown in that figure. An instance of the *MachineryOperationModeStateMachineType* is added to the *Object* MyMachine representing a *Machine* using the *AddIn* concept. The *MachineryOperationModeStateMachineType* is defined in 13.2.

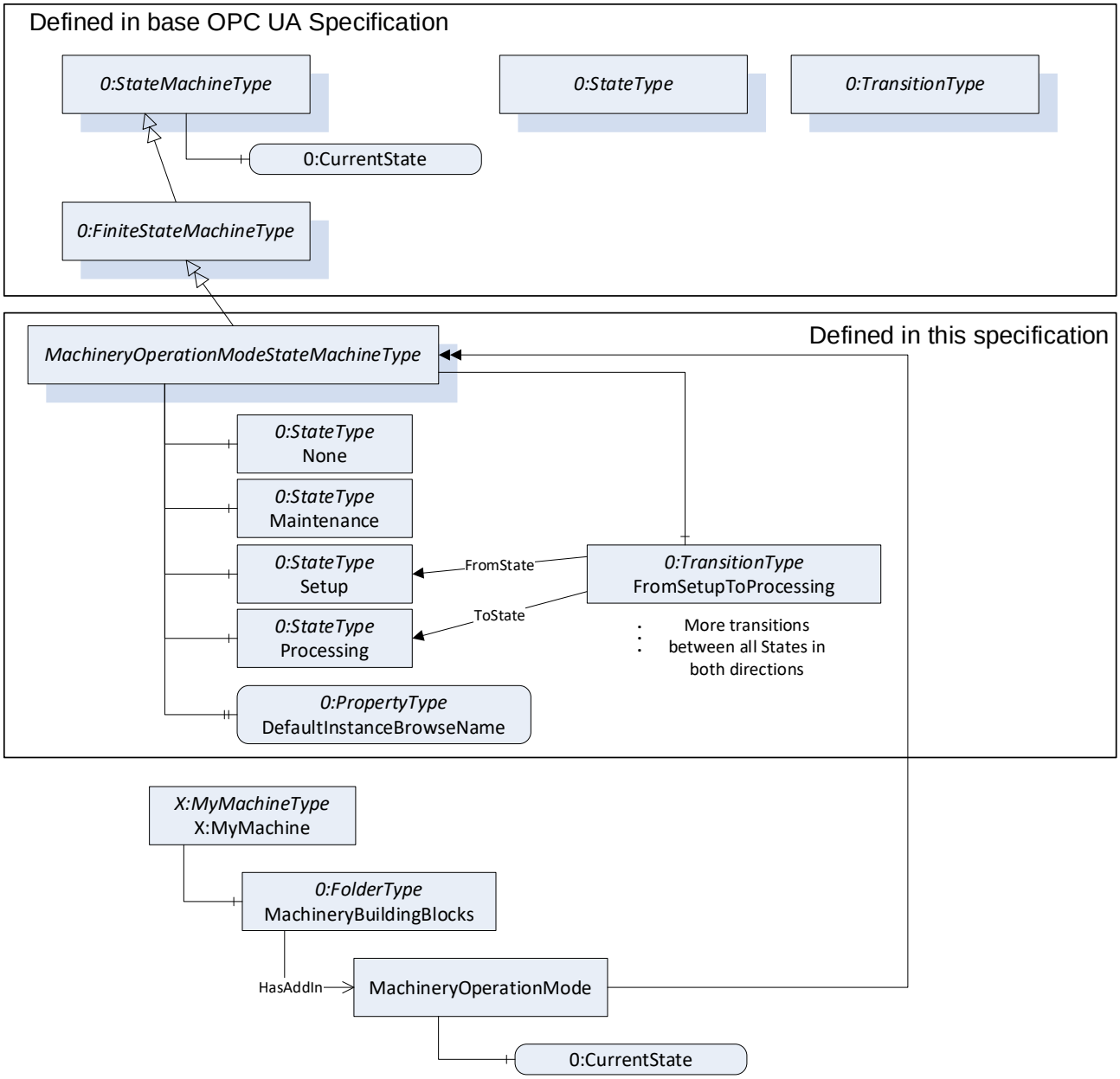

Figure 17 – Building Block for MachineryOperationMode

In Figure 18, a more complex example is shown, where the *Machine* also contains a component having the *MachineryItemState*.

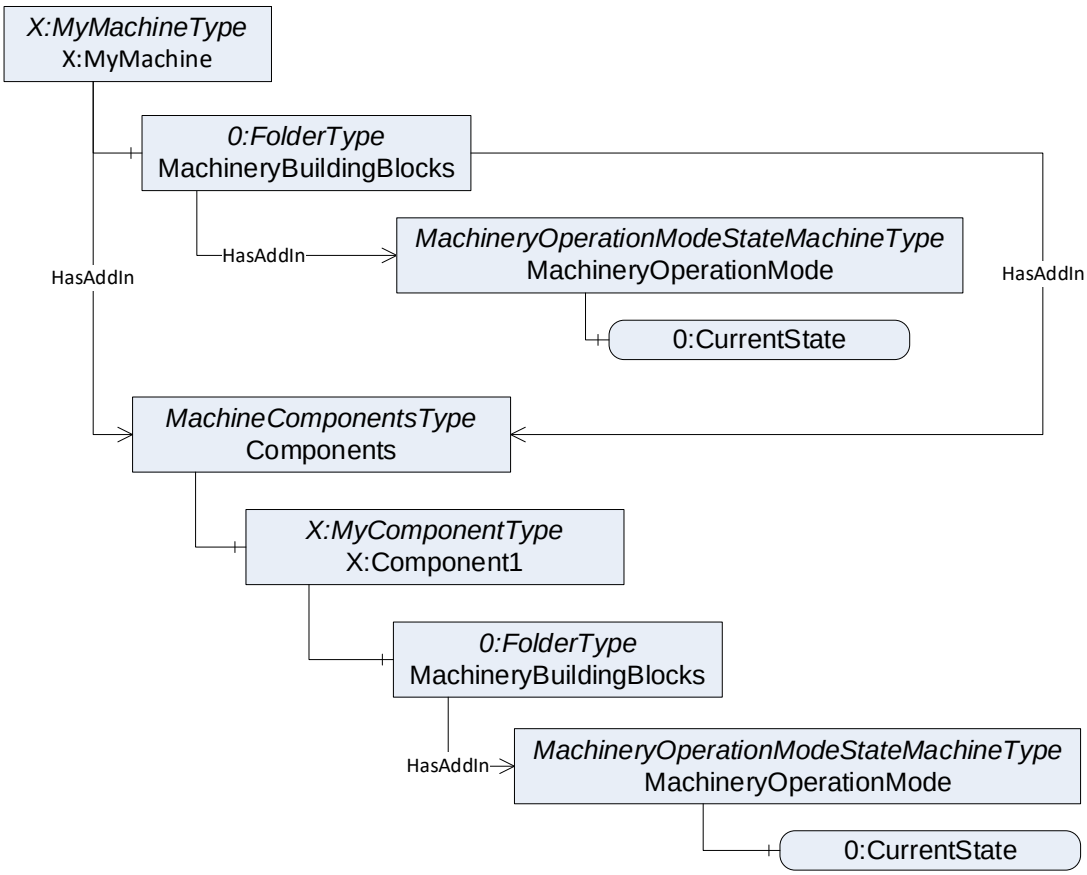

Figure 18 – Example of using the MachineryOperationMode

13.2 MachineryOperationModeStateMachineType

The *MachineryOperationModeStateMachineType* is a subtype of *FiniteStateMachineType* defined in OPC 10000-16. The semantic of this *AddIn* is to provide the *MachineryOperationMode* of a *MachineryItem*. It is formally defined in Table 32.

Table 32 – MachineryOperationModeStateMachineType Definition

Attribute	Value				
BrowseName	MachineryOperationModeStateMachineType				
IsAbstract	False				
Description	State machine representing the operation mode of a MachineryItem				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:FiniteStateMachineType defined in OPC 10000-16, i.e. inheriting the InstanceDeclarations of that Node.					
0:HasProperty	Variable	0:DefaultInstanceBrowseName	0:QualifiedName	0:PropertyType	-
0:HasComponent	Object	None	-	0:StateType	-
0:HasComponent	Object	Maintenance	-	0:StateType	-
0:HasComponent	Object	Processing	-	0:StateType	-
0:HasComponent	Object	Setup	-	0:StateType	-
0:HasComponent	Object	FromNoneToMaintenance	-	0:TransitionType	-
0:HasComponent	Object	FromNoneToSetup	-	0:TransitionType	-
0:HasComponent	Object	FromNoneToProcessing	-	0:TransitionType	-
0:HasComponent	Object	FromNoneToNone	-	0:TransitionType	-
0:HasComponent	Object	FromMaintenanceToNone	-	0:TransitionType	-
0:HasComponent	Object	FromMaintenanceToSetup	-	0:TransitionType	-
0:HasComponent	Object	FromMaintenanceToProcessing	-	0:TransitionType	-
0:HasComponent	Object	FromMaintenanceToMaintenance	-	0:TransitionType	-
0:HasComponent	Object	FromSetupToNone	-	0:TransitionType	-
0:HasComponent	Object	FromSetupToMaintenance	-	0:TransitionType	-
0:HasComponent	Object	FromSetupToProcessing	-	0:TransitionType	-
0:HasComponent	Object	FromSetupToSetup	-	0:TransitionType	-
0:HasComponent	Object	FromProcessingToNone	-	0:TransitionType	-
0:HasComponent	Object	FromProcessingToMaintenance	-	0:TransitionType	-
0:HasComponent	Object	FromProcessingToSetup	-	0:TransitionType	-
0:HasComponent	Object	FromProcessingToProcessing	-	0:TransitionType	-
Conformance Units					
Machinery Operation Mode					

The *States* and *Transitions* define a *StateMachine* as shown in Figure 16. It does not define an initial *State*, i.e., the initial *State* is vendor-specific.

Note: None of the *States* or *Transitions* have a *ModellingRule*, i.e., they are only provided in the *TypeDefinition*, not on the instance. The *CurrentState Variable* (inherited from *StateMachineType*) contains the information of the current *State* of the instance.

The *None State* represents that there is currently no *MachineryOperationMode* available for the *MachineryItem*.

The *Maintenance State* represents that the *MachineryItem* is set into maintenance mode with the intention to carry out maintenance or servicing activities of the *MachineryItem*.

The *Setup State* represents that the *MachineryItem* is set into setup mode with the intention to carry out setup, preparation or postprocessing activities of a production process.

The *Processing State* represents that the *MachineryItem* is set into processing mode with the intention to carry out the value adding activities.

The meaning of the *Transitions* is defined in the *Description* of Table 33.

The *InstanceDeclarations* of the *MachineryOperationModeStateMachineType* have additional *Attribute* values defined in Table 33.

Table 33 – MachineryOperationModeStateMachineType Attribute Values for Child Nodes

SourceBrowsePath	Value	Description
0:DefaultInstanceBrowseName	MachineryOperationMode	The default BrowseName for instances of the type
None	-	There is currently no operation mode available
Maintenance	-	MachineryItem is set into maintenance mode with the intention to carry out maintenance or servicing activities
Setup	-	MachineryItem is set into setup mode with the intention to carry out setup, preparation or postprocessing activities of a production process

Processing	-	MachineryItem is set into processing mode with the intention to carry out the value adding activities
FromNoneToMaintenance	-	Transition from state None to state Maintenance
FromNoneToSetup	-	Transition from state None to state Setup
FromNoneToProcessing	-	Transition from state None to state Processing
FromNoneToNone	-	Transition from state None to state None
FromMaintenanceToNone	-	Transition from state Maintenance to state None
FromMaintenanceToSetup	-	Transition from state Maintenance to state Setup
FromMaintenanceToProcessing	-	Transition from state Maintenance to state Processing
FromMaintenanceToMaintenance	-	Transition from state Maintenance to state Maintenance
FromSetupToNone	-	Transition from state Setup to state None
FromSetupToMaintenance	-	Transition from state Setup to state Maintenance
FromSetupToProcessing	-	Transition from state Setup to state Processing
FromSetupToSetup	-	Transition from state Setup to state Setup
FromProcessingToNone	-	Transition from state Processing to state None
FromProcessingToMaintenance	-	Transition from state Processing to state Maintenance
FromProcessingToSetup	-	Transition from state Processing to state Setup
FromProcessingToProcessing	-	Transition from state Processing to state Processing
None 0:StateNumber	0	-
Maintenance 0:StateNumber	1	-
Setup 0:StateNumber	2	-
Processing 0:StateNumber	3	-
FromNoneToMaintenance 0:TransitionNumber	0	-
FromNoneToProcessing 0:TransitionNumber	1	-
FromNoneToSetup 0:TransitionNumber	2	-
FromMaintenanceToNone 0:TransitionNumber	3	-
FromMaintenanceToProcessing 0:TransitionNumber	4	-
FromMaintenanceToSetup 0:TransitionNumber	5	-
FromProcessingToNone 0:TransitionNumber	6	-
FromProcessingToMaintenance 0:TransitionNumber	7	-
FromProcessingToSetup 0:TransitionNumber	8	-
FromSetupToNone 0:TransitionNumber	9	-
FromSetupToMaintenance 0:TransitionNumber	10	-
FromSetupToProcessing 0:TransitionNumber	11	-
FromNoneToNone 0:TransitionNumber	12	-
FromMaintenanceToMaintenance 0:TransitionNumber	13	-
FromProcessingToProcessing 0:TransitionNumber	14	-
FromSetupToSetup 0:TransitionNumber	15	-

The components of the *MachineryOperationModeStateMachineType* have additional *References* which are defined in Table 34.

Table 34 – MachineryOperationModeStateMachineType Additional References

SourceBrowsePath	Reference Type	Is Forward	TargetBrowsePath
FromNoneToMaintenance	0:FromState	True	None
	0:ToState	True	Maintenance
FromNoneToProcessing	0:FromState	True	None
	0:ToState	True	Processing
FromNoneToSetup	0:FromState	True	None
	0:ToState	True	Setup
FromMaintenanceToNone	0:FromState	True	Maintenance
	0:ToState	True	None
FromMaintenanceToProcessing	0:FromState	True	Maintenance
	0:ToState	True	Processing
FromMaintenanceToSetup	0:FromState	True	Maintenance
	0:ToState	True	Setup
FromProcessingToNone	0:FromState	True	Processing
	0:ToState	True	None
FromProcessingToMaintenance	0:FromState	True	Processing
	0:ToState	True	Maintenance
FromProcessingToSetup	0:FromState	True	Processing
	0:ToState	True	Setup
FromSetupToNone	0:FromState	True	Setup
	0:ToState	True	None
FromSetupToMaintenance	0:FromState	True	Setup
	0:ToState	True	Maintenance
FromSetupToProcessing	0:FromState	True	Setup
	0:ToState	True	Processing
FromNoneToNone	0:FromState	True	None
	0:ToState	True	None
FromMaintenanceToMaintenance	0:FromState	True	Maintenance
	0:ToState	True	Maintenance
FromProcessingToProcessing	0:FromState	True	Processing
	0:ToState	True	Processing
FromSetupToSetup	0:FromState	True	Setup
	0:ToState	True	Setup

14 Operation Counter

14.1 Overview

This building block provides the information, how long a *MachineryItem* is turned on and how long it performed an activity. It uses the 2:*IOperationCounterType* interface and the predefined functional group 2:*OperationCounters* defined in OPC 10000-100.

In Figure 19, an example of a machine providing that information is given. The *MachineryOperationCounterType* defined in 14.2 implements the 2:*IOperationCounterType* interface and provides the 2:*PowerOnDuration* and 2:*OperationDuration Properties*. The X:SpecificMachineType provides the building block in its *MachineryBuildingBlocks* folder using the default name 2:*OperationCounters*. In this case, the X:SpecificMachineType also references the building block directly (like 2:*Identification*), in order to simplify the usage for *Clients* purely developed based on OPC 10000-100. It is recommended to provide this reference, but not required from perspective of this specification.

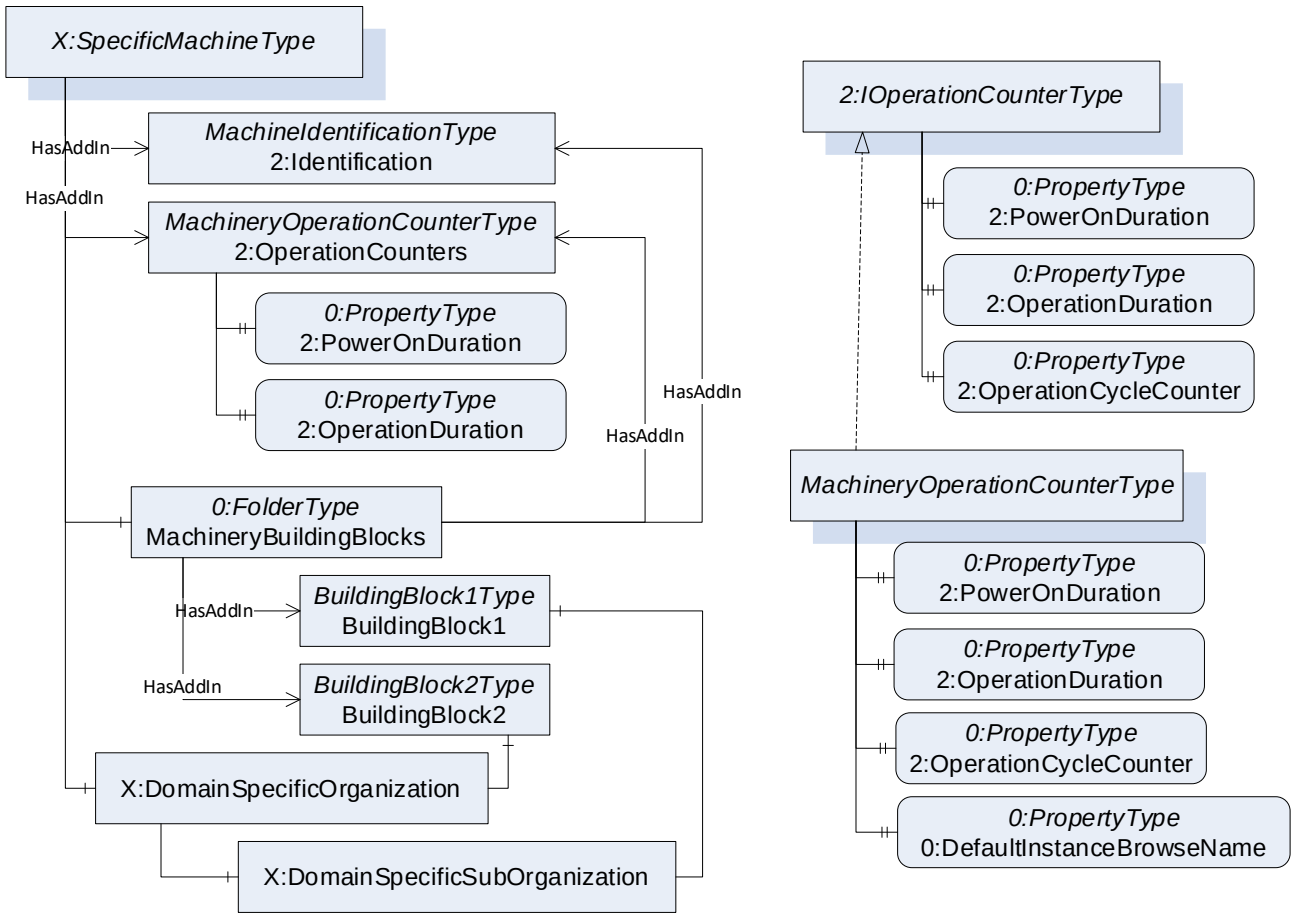

Figure 19 – Example of Building Block for Operation Counter

In Figure 20, an example of a machine and a component of the machine, both providing operation counters, is given. This example leaves out any other building blocks like identification or domain-specific organizations. Like the machine, also the component provides the operation counters.

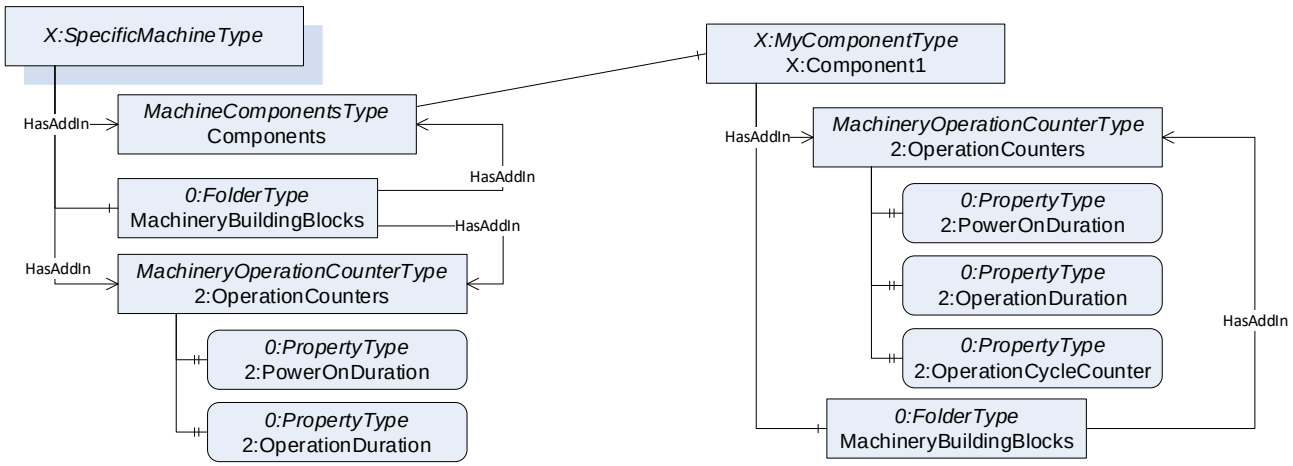

Figure 20 – Example of Building Block for Operation Counter including a component

14.2 MachineryOperationCounterType

The *MachineryOperationCounterType* provides information about the duration something is turned on and how long it performs an activity. It is formally defined in Table 35.

Table 35 – MachineryOperationCounterType Definition

Attribute	**Value**				
BrowseName	MachineryOperationCounterType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 2:FunctionalGroupType defined in OPC 10000-100					
0:HasProperty	Variable	2:PowerOnDuration	0:Duration	0:PropertyType	O
0:HasProperty	Variable	2:OperationDuration	0:Duration	0:PropertyType	O
0:HasProperty	Variable	2:OperationCycleCounter	0:UInteger	0:PropertyType	O
0:HasProperty	Variable	0:DefaultInstanceBrowseName	0:QualifiedName	0:PropertyType	
0:HasInterface	ObjectType	2:IOperationCounterType			
Conformance Units					
Machinery Operation Counter					

The optional 2:*PowerOnDuration*, 2:*OperationDuration* and 2:*OperationCycleCounter* shall be used as defined in OPC 10000-100. Typically, a machine only provides the 2:*PowerOnDuration* to indicate how long the machine is powered on, and the 2:*OperationDuration* to indicate how long the machine is performing an activity. Components of a machine may also provide the 2:*OperationCycleCounter* to identify how often the component was switching.

The child *Nodes* of the *MachineryOperationCounterType* have additional *Attribute* values defined in Table 36.

Table 36 – MachineryOperationCounterType Attribute values for child Nodes

BrowsePath	**Value Attribute**	**Description Attribute**
0:DefaultInstanceBrowseName	2:OperationCounters	The default BrowseName for instances of the type
2:PowerOnDuration	-	PowerOnDuration is the duration the MachineryItem has been powered. The main purpose is to determine the time in which degradation of the MachineryItem occurred. The details, when the time is counted, is implementation-specific. Companion specifications might define specific rules. Typically, when the MachineryItem has supply voltage and the main CPU is running, the time is counted. This may include any kind of sleep mode, but may not include pure Wake on LAN. This value shall only increase during the lifetime of the MachineryItem and shall not be reset when it is restarted. The PowerOnDuration is provided as Duration, i.e., in milliseconds or even fractions of a millisecond. However, the Server is not expected to update the value in such a high frequency, but maybe once a minute or once an hour, depending on the application.
2:OperationDuration	-	OperationDuration is the duration the MachineryItem has been powered and performing an activity. This counter is intended for machines and components where a distinction is made between switched on and in operation. For example, a drive might be powered on but not operating. It is not intended for machines or components always performing an activity like sensors always measuring data. This value shall only increase during the lifetime of the MachineryItem and shall not be reset when it is restarted. The OperationDuration is provided as Duration, i.e., in milliseconds or even fractions of a millisecond. However, the Server is not expected to update the value in such a high frequency, but maybe once a minute or once an hour, depending on the application.
2:OperationCycleCounter	-	OperationCycleCounter is counting the times the component switches from not performing an activity to performing an activity. For example, each time a valve starts moving, is counted. This value shall only increase during the lifetime of the component and shall not be reset when the component is restarted.

15 Lifetime Counter

15.1 Overview

This building block provides the information about the past and estimated remaining lifetime of a *MachineryItem* or other aspects of a *MachineryItem* like a software license. It uses the 2:*LifetimeVariableType* defined in OPC 10000-100 as base.

In Figure 21, an example of a machine providing such information is given. The *MachineryLifetimeCounterType,* defined in 15.2, contains one or many *Variables* of 2:*LifetimeVariableType*. The X:SpecificMachineType provides the building block in its *MachineryBuildingBlocks* folder using the default name *LifetimeCounters*. In the example, there is only one lifetime counter for the validity of a software license, which is also represented as *Object* in the *AddressSpace*.

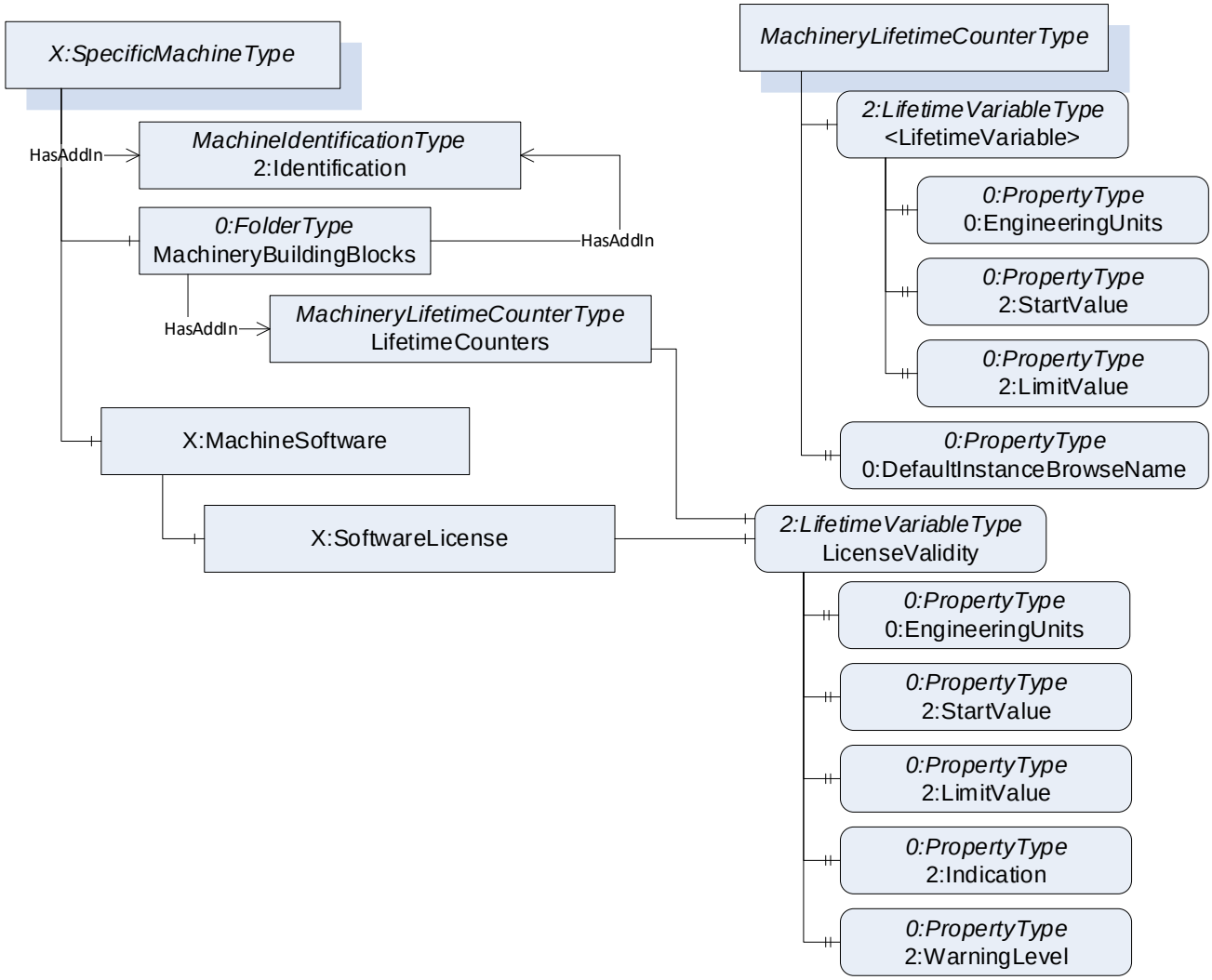

Figure 21 – Example of Building Block for Lifetime Counter

In Figure 22, an example of a machine and a component of the machine, both providing lifetime counters, is given. This example leaves out any other building blocks like identification. Like the Figure 21, the machine provides a lifetime counter for its software. In addition, the component provides a lifetime counter on how many parts it has and can produce. In this case, machine and component provide the LifetimeCounters building block and both reference the PartsProduced lifetime counter, as it is important for both. This is not required, and it is also allowed that the LifetimeCounters of the machine does not reference the PartsProduced or the component does not provide the LifetimeCounters building block.

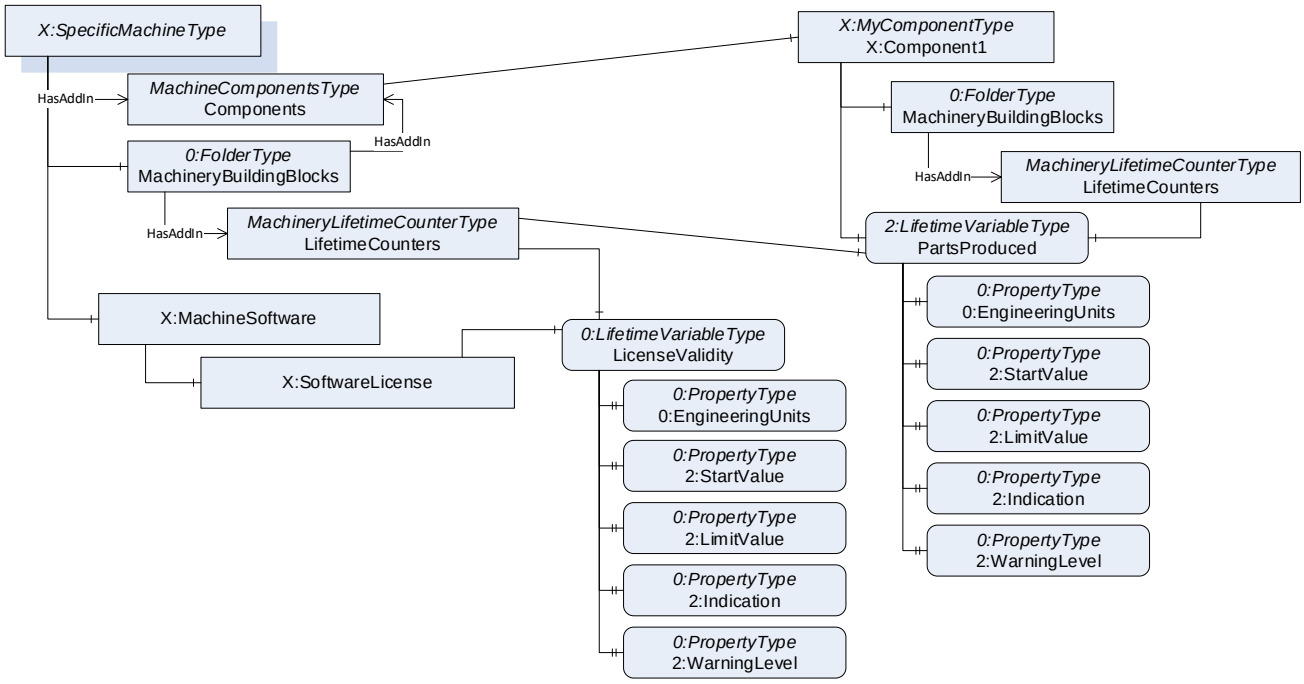

Figure 22 – Example of Building Block for Lifetime Counter including a component

In Table 37, some values are shown that could be the current state of values of the example. The software license has a validity of 365 days, where 200 days are remaining. The parts that can be produced by the component are 1000, where 553 have already been produced.

Table 37 – Lifetime examples

Node	LicenseValidity	PartsProduced
LifetimeVariable	200	553
EngineeringUnits (Mandatory)	day	number of parts
StartValue (Mandatory)	365	0
WarningValues (Optional)	10	950
LimitValue (Mandatory)	0	1000
Indication (Optional)	TimeIndicationType	NumberOfPartsIndicationType

As shown in the examples, it is expected that the lifetime counters can also be accessed by different paths than the building block. The building block allows an optimized access to all relevant counters and thus clients can supervise all those counters and indicate when a warning level or the limit value is reached.

15.2 MachineryLifetimeCounterType

The *MachineryLifetimeCounterType* serves as *AddIn* and provides an entry point to various lifetime variables. It is formally defined in Table 38.

Table 38 – MachineryLifetimeCounterType Definition

Attribute	Value				
BrowseName	MachineryLifetimeCounterType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:FolderType defined in OPC 10000-5					
0:HasComponent	Variable	<LifetimeVariable>	0:Number	2:LifetimeVariableType	MP
0:HasProperty	Variable	0:DefaultInstanceBrowseName	0:QualifiedName	0:PropertyType	
Conformance Units					
Machinery Lifetime Counter					

The <*LifetimeVariable*> can be used for any kind of lifetime variables. There shall be at least one lifetime variable. In situations, where there may be no lifetime variable available, the *AddIn* should be defined as optional on the *TypeDefinition* and should not be provided on the instance if there is no lifetime variable.

The child *Nodes* of the *MachineryLifetimeCounterType* have additional *Attribute* values defined in Table 39.

Table 39 – MachineryLifetimeCounterType Attribute values for child Nodes

BrowsePath	Value Attribute	Description Attribute
0:DefaultInstanceBrowseName	LifetimeCounters	The default BrowseName for instances of the type

16 Profiles and Conformance Units

16.1 Conformance Units

This chapter defines the corresponding *Conformance Units* for the OPC UA Information Model for Machinery.

Table 40 – Conformance Units for OPC UA for Machinery

Category	Title	Description
Server	Machinery Machine Identification	Supports the MachineIdentificationType with all its mandatory InstanceDeclarations, and optionally the optional InstanceDeclarations with read access. There is at least one instance of the MachineIdentificationType or a subtype using the DefaultInstanceBrowseName and is referenced from an Object representing a Machine with a Reference of HasAddIn or a subtype.
Server	Machinery Machine Identification Writable	Supports the MachineIdentificationType with all its mandatory InstanceDeclarations, and optionally the optional InstanceDeclarations, with writable access to all Variables defined as writable in this specification. The optional Properties 2:AssetId, 2:ComponentName and Location shall be provided for all instances of the MachineIdentificationType or its subtypes.
Server	Machinery Find Machines	Supports the Machines Object and references all Machines of the Server as defined by the Machines Object.
Server	Machinery Component Identification	Supports the MachineryComponentIdentificationType with optionally the optional InstanceDeclarations with read access. There is at least one instance of the MachineryComponentIdentificationType or a subtype using the DefaultInstanceBrowseName and is referenced from an Object representing a component of a Machine with a Reference of HasAddIn or a subtype.
Server	Machinery Component Identification Mandatory	Supports the MachineryComponentIdentificationType with the Properties 2:Manufacturer, 2:Model, 2:ProductCode, and 2:SerialNumber as mandatory and optionally the other optional InstanceDeclarations with read access. The Properties 2:Manufacturer, 2:Model, 2:ProductCode, and 2:SerialNumber shall be provided for all instances of the MachineryComponentIdentificationType or a subtype.
Server	Machinery Component Identification Writable	Supports the MachineryComponentIdentificationType with optionally the optional InstanceDeclarations, with writable access to all Variables defined as writable in this specification for all instances of the MachineryComponentIdentificationType or its subtypes. The optional Properties 2:AssetId, and 2:ComponentName shall be provided for all instances of the MachineryComponentIdentificationType or its subtypes.
Server	Machinery Find Components of Machines	Supports the MachineComponentsType for all Machines managed by the Server, each one referencing the exposed components of the corresponding machine. Each Object representing a Machine shall reference an instance of MachineComponentsType or a subtype using the DefaultInstanceBrowseName with a Reference of HasAddIn or a subtype.
Server	Machinery Building Block Organization	Each MachineryItem supporting building blocks defined by this specification using the MachineryBuildingBlocks folder directly references a folder called MachineryBuildingBlocks. This folder directly references all those building blocks supported by the MachineryItem.
Server	Machinery MachineryItem State	Supports the MachineryItemState_StateMachineType and has at least one instance of a MachineryItem supporting this as AddIn under its MachineryBuildingBlocks folder.
Server	Machinery Operation Mode	Supports the MachineryOperationModeStateMachineType and has at least one instance of a MachineryItem supporting this as AddIn under its MachineryBuildingBlocks folder.
Server	Machinery Operation Counter	Supports the MachineryOperationCounterType and has at least one instance of a MachineryItem supporting this as AddIn with at least one counter under its MachineryBuildingBlocks folder.
Server	Machinery Lifetime Counter	Supports the MachineryLifetimeCounterType and has at least one instance of a MachineryItem supporting this as AddIn under its MachineryBuildingBlocks folder.

16.2 Profiles

16.2.1 Profile list

Table 41 lists all *Profiles* defined in this document and defines their URIs.

Table 41 – Profile URIs for OPC UA for Machinery

Profile	URI
Machinery Machine Identification Server Facet	http://opcfoundation.org/UA-Profile/Machinery/Server/MachineIdentification
Machinery Machine Identification Writable Server Facet	http://opcfoundation.org/UA-Profile/Machinery/Server/MachineIdentificationWritable
Machinery Component Identification Server Facet	http://opcfoundation.org/UA-Profile/Machinery/Server/ComponentIdentification
Machinery Component Identification Mandatory Server Facet	http://opcfoundation.org/UA-Profile/Machinery/Server/ComponentIdentificationMandatory
Machinery Component Identification Writable Server Facet	http://opcfoundation.org/UA-Profile/Machinery/Server/ComponentIdentificationWritable
Machinery State Server Facet	http://opcfoundation.org/UA-Profile/Machinery/Server/State
Machinery Operation Counter Server Facet	http://opcfoundation.org/UA-Profile/Machinery/Server/OperationCounter
Machinery Lifetime Counter Server Facet	http://opcfoundation.org/UA-Profile/Machinery/Server/LifetimeCounter

16.2.2 Server Facets

16.2.2.1 Overview

The following sections specify the *Facets* available for *Servers* that implement the OPC UA for Machinery companion specification. Each section defines and describes a *Facet* or *Profile*.

16.2.2.2 Machinery Machine Identification Server Facet

Table 42 defines a *Facet* that provides the identification of *Machines* managed in an OPC UA Server.

Table 42 – Machinery Machine Identification Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Address Space Model	0:Address Space Base	M
Address Space Model	0:Address Space Interfaces	M
Address Space Model	0:Address Space AddIn Reference	M
Address Space Model	0:Address Space AddIn DefaultInstanceBrowsename	M
View Services	0:View Basic	M
View Services	0:View TranslateBrowsePath	M
View Services	0:View Minimum Continuation Point 01	M
Attribute Services	0:Attribute Read	M
Machinery	Machinery Machine Identification	M
Machinery	Machinery Find Machines	M
Machinery	Machinery Building Block Organization	O

16.2.2.3 Machinery Machine Identification Writable Server Facet

Table 43 defines a *Facet* that provides the identification of *Machines* as well as the writing of *Machine* identification aspects changeable by the user via the OPC UA interface.

Table 43 – Machinery Machine Identification Writable Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Profile	Machinery Machine Identification Server Facet	
Attribute Services	0:Attribute Write Values	M
Machinery	Machinery Machine Identification Writable	M

16.2.2.4 Machinery Component Identification Server Facet

Table 44 defines a *Facet* that provides the identification of components of *Machines* managed in an OPC UA Server.

Table 44 – Machinery Component Identification Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Address Space Model	0:Address Space Base	M
Address Space Model	0:Address Space Interfaces	M
Address Space Model	0:Address Space AddIn Reference	M
Address Space Model	0:Address Space AddIn DefaultInstanceBrowsename	M
View Services	0:View Basic	M
View Services	0:View TranslateBrowsePath	M
View Services	0:View Minimum Continuation Point 01	M
Attribute Services	0:Attribute Read	M
Machinery	Machinery Component Identification	M
Machinery	Machinery Find Components of Machines	M
Machinery	Machinery Building Block Organization	O

16.2.2.5 Machinery Component Identification Mandatory Server Facet

Table 45 defines a *Facet* that provides the identification of components of *Machines* with some mandatory Properties via the OPC UA interface.

Table 45 – Machinery Component Identification Mandatory Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Profile	Machinery Component Identification Server Facet	
Machinery	Machinery Component Identification Mandatory	M

16.2.2.6 Machinery Component Identification Writable Server Facet

Table 46 defines a *Facet* that provides the identification of *Machines* as well as the writing of *Machine* identification aspects changeable by the user via the OPC UA interface.

Table 46 – Machinery Component Identification Writable Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Profile	Machinery Component Identification Server Facet	
Attribute Services	0:Attribute Write Values	M
Machinery	Machinery Component Identification Writable	M

16.2.2.7 Machinery State Server Facet

Table 47 defines a *Facet* that provides the states and modes of *MachineryItems* managed in an OPC UA Server.

Table 47 – Machinery State Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Address Space Model	0:Address Space Base	M
Address Space Model	0:Address Space AddIn Reference	M
Address Space Model	0:Address Space AddIn DefaultInstanceBrowsename	M
View Services	0:View Basic	M
View Services	0:View TranslateBrowsePath	M
View Services	0:View Minimum Continuation Point 01	M
Attribute Services	0:Attribute Read	M
Machinery	Machinery Building Block Organization	M
Machinery	Machinery MachineryItem State	M
Machinery	Machinery Operation Mode	O

16.2.2.8 Machinery Operation Counter Server Facet

Table 48 defines a *Facet* that provides operation counter of *MachineryItems* managed in an OPC UA Server.

Table 48 – Machinery Operation Counter Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Address Space Model	0:Address Space Base	M
Address Space Model	0:Address Space AddIn Reference	M
Address Space Model	0:Address Space AddIn DefaultInstanceBrowsename	M
View Services	0:View Basic	M
View Services	0:View TranslateBrowsePath	M
View Services	0:View Minimum Continuation Point 01	M
Attribute Services	0:Attribute Read	M
Machinery	Machinery Building Block Organization	M
Machinery	Machinery Operation Counter	M

16.2.2.9 Machinery Lifetime Counter Server Facet

Table 49 defines a *Facet* that provides lifetime counter of *MachineryItems* managed in an OPC UA Server.

Table 49 – Machinery Lifetime Counter Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Address Space Model	0:Address Space Base	M
Address Space Model	0:Address Space AddIn Reference	M
Address Space Model	0:Address Space AddIn DefaultInstanceBrowsename	M
View Services	0:View Basic	M
View Services	0:View TranslateBrowsePath	M
View Services	0:View Minimum Continuation Point 01	M
Attribute Services	0:Attribute Read	M
Machinery	Machinery Building Block Organization	M
Machinery	Machinery Lifetime Counter	M

16.2.3 Client Facets

This version of the specification does not define any *Client Facets*.

17 Namespaces

17.1 Namespace Metadata

Table 50 defines the namespace metadata for this document. The *Object* is used to provide version information for the namespace and an indication about static *Nodes*. Static *Nodes* are identical for all *Attributes* in all *Servers*, including the *Value Attribute*. See OPC 10000-5 for more details.

The information is provided as *Object* of type *NamespaceMetadataType*. This *Object* is a component of the *Namespaces Object* that is part of the *Server Object*. The *NamespaceMetadataType ObjectType* and its *Properties* are defined in OPC 10000-5.

The version information is also provided as part of the ModelTableEntry in the UANodeSet XML file. The UANodeSet XML schema is defined in OPC 10000-6.

Table 50 – NamespaceMetadata Object for this Document

Attribute	Value	
BrowseName	http://opcfoundation.org/UA/Machinery/	
Property	**DataType**	**Value**
NamespaceUri	String	http://opcfoundation.org/UA/Machinery/
NamespaceVersion	String	1.03.0
NamespacePublicationDate	DateTime	2023-08-01
IsNamespaceSubset	Boolean	False
StaticNodeIdTypes	IdType[]	0
StaticNumericNodeIdRange	NumericRange []	-
StaticStringNodeIdPattern	String	-

Note: The *IsNamespaceSubset Property* is set to False as the UaNodeSet XML file contains the complete Namespace. *Servers* only exposing a subset of the Namespace need to change the value to True.

17.2 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 *AddressSpace* is unambiguously identified using a *NodeId*. Unlike *NodeIds*, the *BrowseName* cannot be used to unambiguously identify a *Node*. Different *Nodes* may have the same *BrowseName*. They are used to build a browse path between two *Nodes* or to define a standard *Property*.

Servers may often choose to use the same namespace for the *NodeId* and the *BrowseName*. However, if they want to provide a standard *Property*, its *BrowseName* 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 document shall not use the standard namespaces.

Table 51 provides a list of mandatory and optional namespaces used in an OPC UA for Machinery OPC UA *Server*.

Table 51 – Namespaces used in an OPC UA for Machinery Server

NamespaceURI	Description	Use
http://opcfoundation.org/UA/	Namespace for *NodeIds* and *BrowseNames* 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 namespace shall have namespace index 1.	Mandatory
http://opcfoundation.org/UA/DI/	Namespace for NodeIds and BrowseNames defined in OPC 10000-100. The namespace index is Server specific.	Mandatory
http://opcfoundation.org/UA/Machinery/	Namespace for *NodeIds* and *BrowseNames* defined in this document. The namespace index is *Server* specific.	Mandatory
Vendor specific types	A *Server* may provide vendor-specific types like types derived from *ObjectTypes* defined in this document in a vendor-specific namespace.	Optional
Vendor specific instances	A *Server* provides vendor-specific instances of the standard types or vendor-specific instances of vendor-specific types in a vendor-specific namespace. It is recommended to separate vendor specific types and vendor specific instances into two or more namespaces.	Mandatory

Table 52 provides a list of namespaces and their index used for *BrowseNames* in this document. The default namespace of this document is not listed since all *BrowseNames* without prefix use this default namespace.

Table 52 – Namespaces used in this Document

NamespaceURI	Namespace Index	Example
http://opcfoundation.org/UA/	0	0:EngineeringUnits
http://opcfoundation.org/UA/DI/	2	2:DeviceRevision

Annex A
(normative)

OPC UA for Machinery Namespace and Mappings

A.1 Namespace and identifiers for Machinery Information Model

This appendix defines the numeric identifiers for all of the numeric *NodeIds* defined in this specification. The identifiers are specified in a CSV file with the following syntax:

`<SymbolName>, <Identifier>, <NodeClass>`

Where the *SymbolName* is either the *BrowseName* of a *Type Node* or the *BrowsePath* for an *Instance Node* that appears in the specification and the *Identifier* is the numeric value for the *NodeId*.

The *BrowsePath* for an *Instance Node* is constructed by appending the *BrowseName* of the instance *Node* to the *BrowseName* for the containing instance or type. An underscore character is used to separate each *BrowseName* in the path. Let's take for example, the *MachineIdentificationType ObjectType Node* which has the *InitialOperationDate Property*. The **Name** for the *InitialOperationDate InstanceDeclaration* within the *MachineIdentificationType* declaration is: *MachineIdentificationType_InitialOperationDate*.

The *NamespaceUri* for all *NodeIds* defined here is http://opcfoundation.org/UA/Machinery/

The CSV released with this version of the specification can be found here:

– http://www.opcfoundation.org/UA/schemas/Machinery/1.02/NodeIds.csv

NOTE The latest CSV that is compatible with this version of the specification can be found here:

– http://www.opcfoundation.org/UA/schemas/Machinery/NodeIds.csv

A computer processible version of the complete Information Model defined in this specification is also provided. It follows the XML Information Model schema syntax defined in OPC 10000-6.

The Information Model Schema for this version of the document can be found here:

– http://www.opcfoundation.org/UA/schemas/Machinery/1.02/Opc.Ua.Machinery.NodeSet2.xml

NOTE The latest Information Model schema that is compatible with this version of the specification can be found here:

– http://www.opcfoundation.org/UA/schemas/Machinery/Opc.Ua.Machinery.NodeSet2.xml

_____

Annex B
(informative)

Examples

B.1 Overview

This appendix provides informal examples on how the building blocks defined in this specification can be used.

B.2 Identification and Finding Machines

In Figure 23, an example is given, showing the identification and Nameplate and Finding all *Machines* in Server use cases. The server provides information about a Robotics system as well as a CNC machine defined by Machine Tools (see OPC 40501, [3]). As the Robotics specification (see OPC 40010-1, [4]) already defines some *Properties* for identification directly, those are only referenced from the Identification functional group.

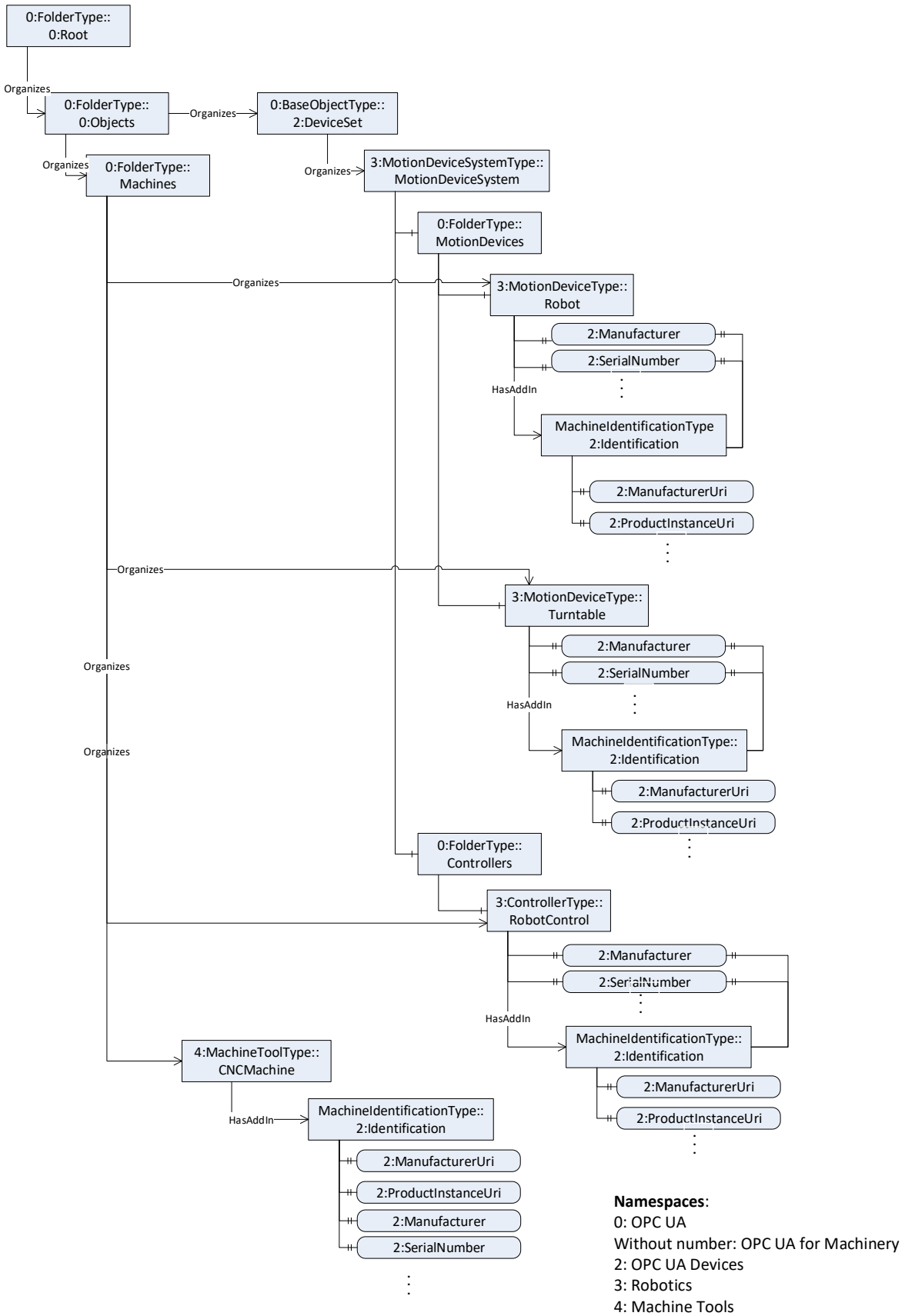

Figure 23 – Example of Identification

B.3 Component Identification and Finding Components of a Machine

In Figure 24, an example is given, showing the Component Identification and Nameplate and Finding all Components of a *Machine* use cases. The server provides information about a Robotics system (see OPC 40010-1, [4]). In that system, the components are organized according to the Robotics specification, and in addition according to this specification.

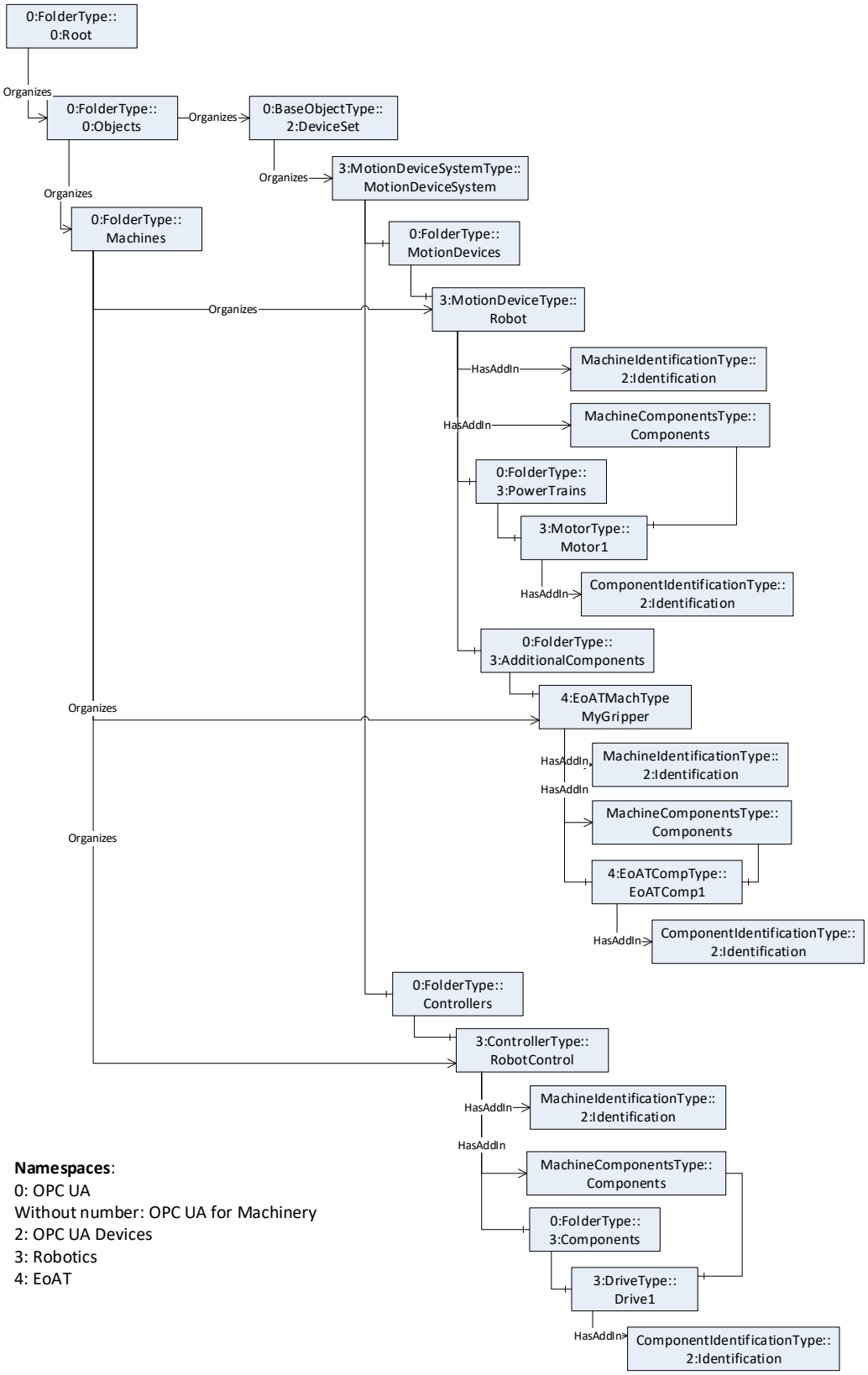

Figure 24 – Example of Component Identification

Annex C
(informative)

Examples how to use MachineryItemState and MachineryOperationMode to calculate KPIs

C.1 Overview

To calculate KPIs such as the availability or efficiency of a machine or plant, the respective times must be used. These can be derived from the *MachineryItemState* and the *MachineryOperationMode*. This annex does not define any KPI values. The assumption is, that the KPIs will be calculated application specific, but can be based on those two states. Potentially, more information is needed for calculating specific KPIs.

To calculate KPI often information about the planning of production is required. This information usually comes from a higher level system such as MES or ERP and not from the machine, so it has not been included in this specification.

The following table in C.2 is used to explain how the required KPI time elements can be derived conform to ISO 22400-2 using the combination of *MachineryItemState*, *MachineryOperationMode* and HLS-ConditionState of the higher level system such as MES or ERP on level 3 (see IEC 62264-1). The KPI element definitions used are taken from the ISO 22400-2:2022 called Automation systems and integration - Key performance indicators (KPIs) for manufacturing operations management for KPI calculation.

The following section C.3 provides an example interpretation, of how the *MachineryOperationMode* and *MachineryItemState* can be used to determine SEMI E10 [5] times. Typically, *MachineryItems* are scheduled by higher level systems and are not aware how they are scheduled. So, interpretations of SEMI E10 times are typically “Operations time”.

C.2 Example interpretation for ISO 22400

Table 53 proposes how KPI time elements according to ISO 22400-2 can be set using the combinations of *MachineryItemState*, *MachineryOperationMode* and *HLS-ConditionState*. These ISO 22400-2 KPI time elements can be used to do KPI calculations.

Table 53 – KPI time elements according to ISO 22400-2

ISO 22400-2 KPI time elements	**ISO 22400-2 Interpretation**	**Machinery Item State**	**Machinery Operation Mode**	**Higher Level System Condition State**
ADET (Actual Delay Time)	Delay time	OutOfService	Processing	System state is order registered but no maintenance
	Delay time	NotAvailable	Processing (last sent)	System state is order registered but no maintenance
	Delay time	NotExecuting	None	System state is order registered
	Delay time	NotExecuting	Processing	System state is order registered but no maintenance and time is just out of cycle time
ADOT (Actual Down Time)	Down time	OutOfService	None	System state is planned busy time based on operation calendar
	Down time	NotAvailable	None (last sent)	System state is planned busy time based on operation calendar but no order registered
	Down time	NotExecuting	None	System state is planned busy time based on operation calendar but no order registered

ISO 22400-2 KPI time elements	ISO 22400-2 Interpretation	Machinery Item State	Machinery Operation Mode	Higher Level System Condition State
APMT (Actual Preventive Maintenance Time)	preventive maintenance as service	OutOfService	Maintenance	System state is planned operation time based on operation calendar but no order registered
	Setup time as service	OutOfService	Setup	System state is planned operation time based on operation calendar but no order registered
	Setup time as service	NotAvailable	Setup (last sent)	System state is planned operation time based on operation calendar but no order registered
	preventive maintenance as service	NotAvailable	Maintenance (last sent)	System state is planned operation time based on operation calendar but no order registered
	preventive maintenance as service	NotExecuting	Maintenance	System state is planned operation time based on operation calendar but no order registered
	Setup time as service	NotExecuting	Setup	System state is planned operation time based on operation calendar but no order registered
	preventive maintenance as service	Executing	Maintenance	System state is planned operation time based on operation calendar but no order registered
APT (Actual production time)	Production time	Executing	None	System state is order registered
	Production time	Executing	Processing	System state is order registered
	Production time	NotExecuting	Processing	System state is order registered
	Production time	Executing	Setup	SPLIT in setup time (AUST) and value added production time (APT=PQ*RPI) as retrograde confirmation (backflush)
ASDT (Actual Shut Down Time)	Shut down time	OutOfService	None	System state is shut down time based on operation calendar
	Shut down time	NotAvailable	None (last sent)	System state is shut down time based on operation calendar
	Shut down time	NotExecuting	None	System state is shut down time based on operation calendar
AUST (Actual Unit Setup Time)	Setup time	OutOfService	Setup	System state is order registered
	Setup time	NotAvailable	Setup (last sent)	System state is order registered
	Setup time	NotExecuting	Setup	System state is order registered
	Setup time	Executing	Setup	SPLIT in setup time (AUST) and value added production time (APT=PQ*RPI) as retrograde confirmation (backflush)

ISO 22400-2 KPI time elements	ISO 22400-2 Interpretation	Machinery Item State	Machinery Operation Mode	Higher Level System Condition State
TTR (Time to Repair)	Delay time but maintenance	OutOfService	Maintenance	System state is order registered but maintenance activities
	Delay time but maintenance	OutOfService	Processing	System state is order registered but maintenance activities
	Delay time but maintenance	NotAvailable	Processing (last sent)	System state is order registered but maintenance activities
	Delay time but maintenance	NotAvailable	Maintenance (last sent)	System state is order registered but maintenance activities
	Delay time but maintenance	NotExecuting	Maintenance	System state is order registered but maintenance activities
	Delay time but maintenance	NotExecuting	Processing	System state is order registered but maintenance activities and time is out of cycle time
	Delay time but maintenance	Executing	Maintenance	System state is order registered but maintenance activities

Note 1: If condition state is order registered but in planned break down time period based on operation calendar then KPI time element has to be actual break down time (ABRT).

Note 2: Technological effects e.g. speed losses based on wait for feeds or wait for drains have to be analysed on SCADA or machinery level.

*Note 3: If an added value activity is included in setup mode period, then the added value time has to be calculated as APT (APT=PQ*RPI) based on produced quantities and AUST has to be reduced based on this calculated APT.*

C.3 Example interpretation for SEMI E10

The *MachineryItemState OutOfService* and the *MachineryOperationMode Processing* can be interpreted as SEMI E10 “Unscheduled Downtime”.

The *MachineryItemState NotExecuting* and the *MachineryOperationMode Setup* can be interpreted as SEMI E10 “Engineering time”.

The *MachineryItemState NotExecuting* and the *MachineryOperationMode Processing* can be interpreted as SEMI E10 “Standby time”.

The *MachineryItemState Executing* and the *MachineryOperationMode Processing* can be interpreted as SEMI E10 “Productive time”.

In this example, the availability of the *Machine* is calculated according to ISO 22400. For this purpose, the Actual Production Time (APT) and Planned Busy Time (PBT) are required. The APT can be derived from the combination of the *MachineryItemState* "Executing" and the *MachineryOperationMode* "Processing". Alternatively, the APT can also be derived from the Semi E10 time "Productive time". The PBT is usually not known to the *Machine* and is accordingly not provided by this specification.

_____

Bibliography

[1] ISO 12100:2010, *Safety of machinery – General principles for design – Risk assessment and risk reduction*

[2] IEC 61499-1:2012, *Function Blocks – Part 1: Architecture*

[3] OPC 40501-1, *UA for Machine Tools Part 1: Machine Monitoring and Job Overview*

(http://www.opcfoundation.org/UA/MachineTools/)

[4] OPC 40010-1, *OPC UA for Robotics – Part 1: Vertical Integration*

(http://www.opcfoundation.org/UA/Robotics/)

[5] SEMI E10, *Specification for Definition and Measurement of Equipment Reliability, Availability, and Maintainability (RAM) and Utilization*

[6] ISO 22400-2, *Automation systems and integration — Key performance indicators (KPIs) for manufacturing operations management — Part 2: Definitions and descriptions*

———————