VDMA 40001-1

ICS 25.020; 35.240.50

Comments by 2020-09-01

OPC UA for Machinery – Part 1: Basic Building Blocks

OPC UA for Machinery –
Teil 1: Basic Building Blocks

VDMA 40001-1:2020-06 is identical with OPC 40001-1 (Release Candidate 1.0.0)

Application Warning Notice

This draft with date of issue 2020-06-01 is being submitted to the public for review and comment.

Because the final VDMA Specification may differ from this version, the application of this draft is subject to special agreement.

Comments are requestet

– preferably as a file by e-mail to heiko.herden@vdma.org

– or in paper form to VDMA e.V., Forum Industry 4.0, Lyoner Straße 18, 60528 Frankfurt.

Document comprises 33 pages

VDMA

Contents

Figures

Tables

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.

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 3200 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 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
– Finding all Machines in a Server

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 10001-7, *OPC Unified Architecture V1.04 - Amendment 7: Interfaces ad AddIns*

http://www.opcfoundation.org/UA/Amendment7/

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 10000-5, OPC 10000-7, OPC 10000-100, and OPC 100001-7 as well as the following apply.

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

3.2 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
MES	Manufacturing Execution System
OPC UA	Open Platform Communications Unified Architecture
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

3.3 Conventions used in this document

3.3.1 Conventions for Node descriptions

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 3.3.2.2).

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.

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*. If the *Reference* is to be instantiated by the server, then the value of the target Node's BrowseName is "--".	*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.					

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.

3.3.2 NodeIds and BrowseNames

3.3.2.1 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*.

3.3.2.2 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.

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 22 provides a list of namespaces and their indexes as used in this document.

3.3.3 Common Attributes

3.3.3.1 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 4 shall be set as specified in the table.

Table 4 – 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 3.3.2.1.
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-specifc and depend on the *RolePermissions Attribute* (if provided) and the current *Session*.
AccessRestrictions	Optionally server-specific access restrictions can be provided.

3.3.3.2 Objects

For all *Objects* specified in this specification, the *Attributes* named in Table 5 shall be set as specified in the table. The definitions for the *Attributes* can be found in OPC 10000-3.

Table 5 – Common Object Attributes

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

3.3.3.3 Variables

For all *Variables* specified in this specification, the *Attributes* named in Table 6 shall be set as specified in the table. The definitions for the *Attributes* can be found in OPC 10000-3.

Table 6 – 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.

3.3.3.4 VariableTypes

For all *VariableTypes* specified in this specification, the *Attributes* named in Table 7 shall be set as specified in the table. The definitions for the *Attributes* can be found in OPC 10000-3.

Table 7 – 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*.

3.3.3.5 Methods

For all *Methods* 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 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.

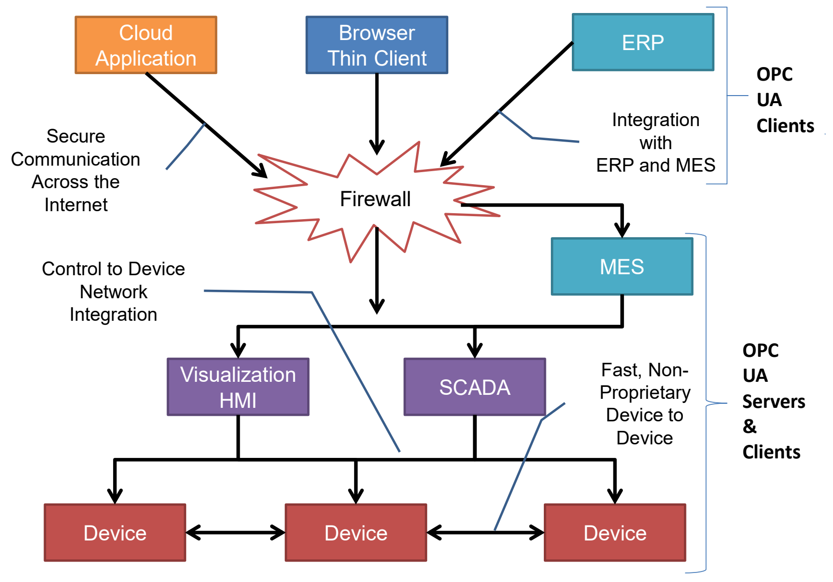
Cloud Application
Browser Thin Client
ERP
OPC UA Clients
Secure Communication Across the Internet
Integration with ERP and MES
Firewall
MES
Control to Device Network Integration
Visualization HMI
SCADA
Fast, Non-Proprietary Device to Device
OPC UA Servers & Clients
Device
Device
Device

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.

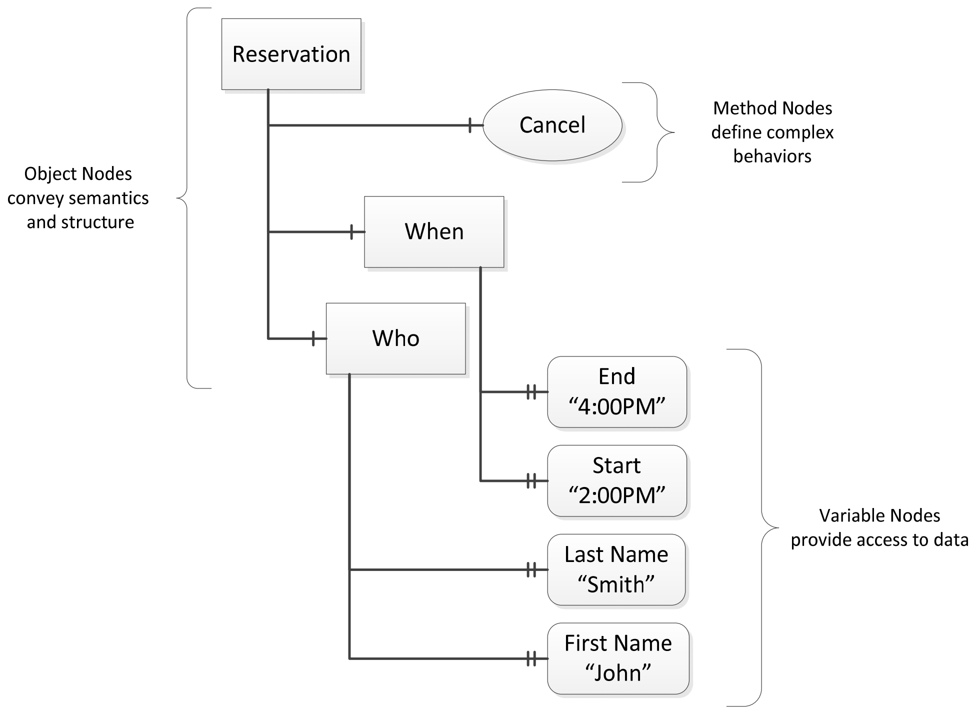
Reservation
Cancel
Method Nodes define complex behaviors
Object Nodes convey semantics and structure
When
Who
End “4:00PM”
Start “2:00PM”
Last Name “Smith”
First Name “John”
Variable Nodes provide access to data

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. 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 sub-typing. This allows a modeller to take an existing type and extend it. There are rules regarding sub-typing 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.

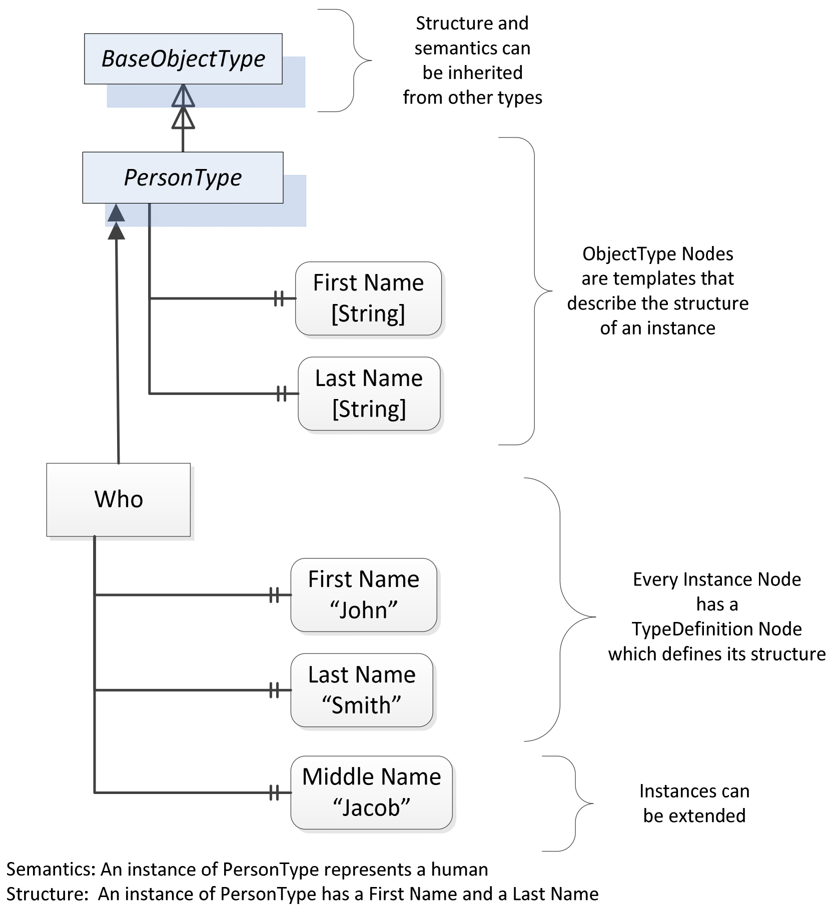
BaseObjectType
Structure and semantics can be inherited from other types
PersonType
First Name [String]
Last Name [String]
ObjectType Nodes are templates that describe the structure of an instance
Who
First Name "John"
Last Name "Smith"
Every Instance Node has a TypeDefinition Node which defines its structure
Middle Name "Jacob"
Instances can be extended
Semantics: An instance of PersonType represents a human
Structure: An instance of PersonType has a First Name and a Last Name

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*.

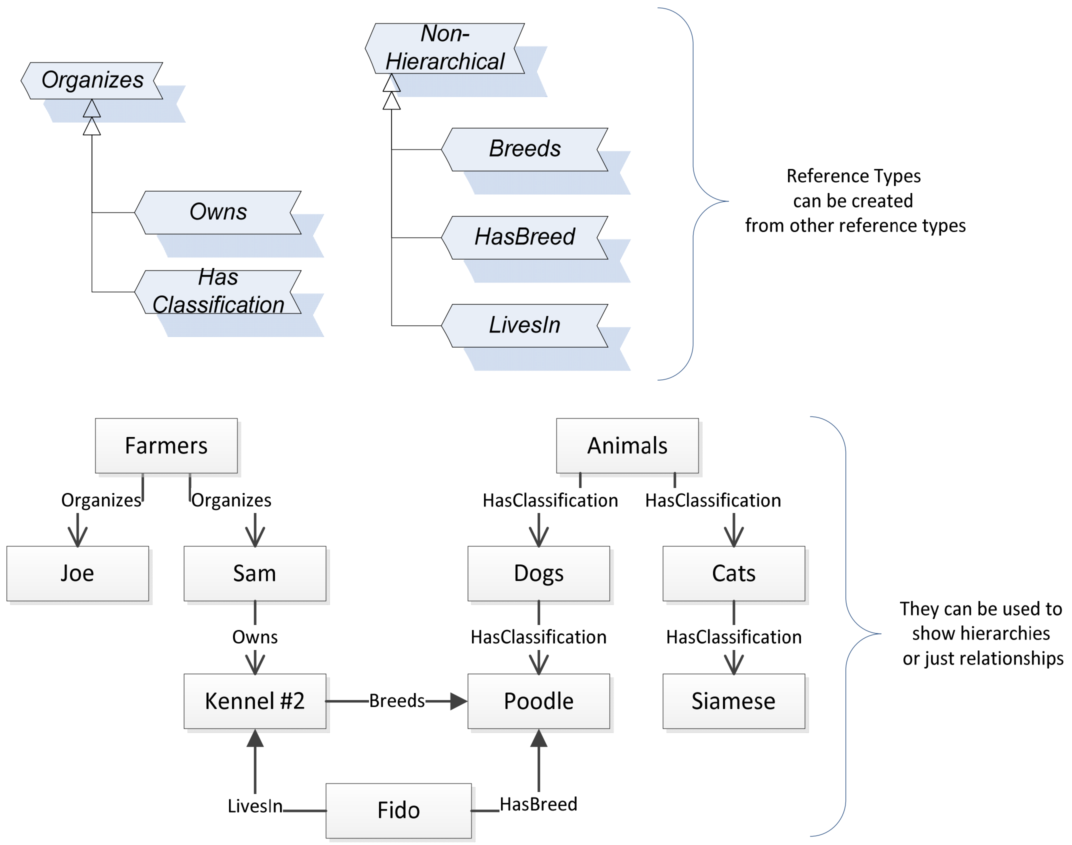

Figure 4 – Examples of References between Objects

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*.

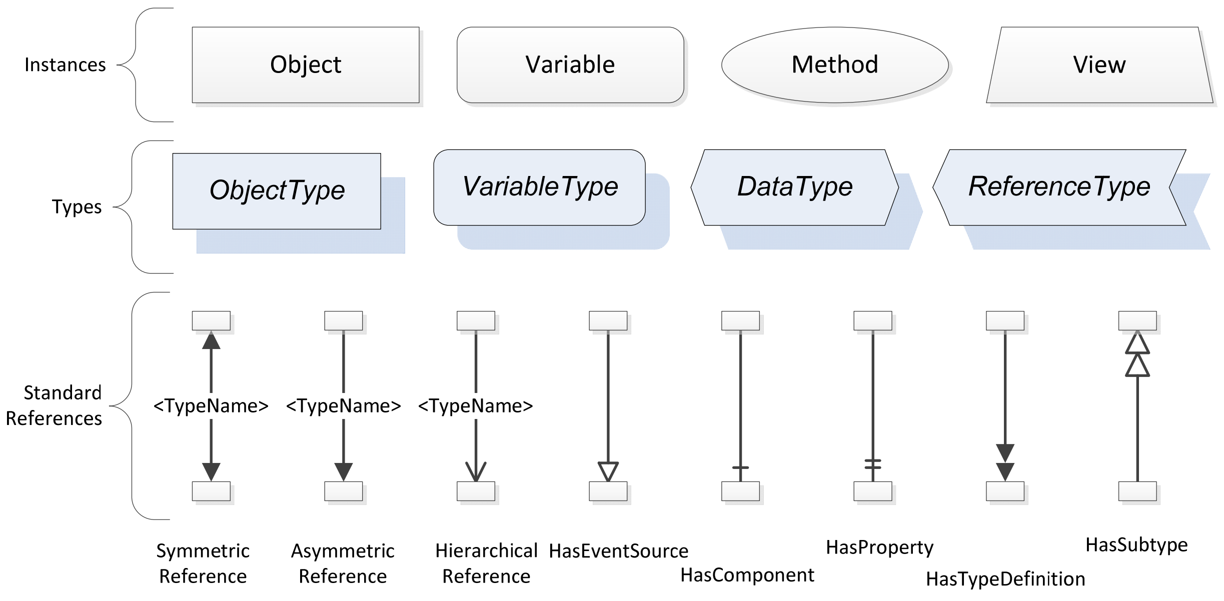

Figure 5 – The OPC UA Information Model Notation

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 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*).

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 10001-7, 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.

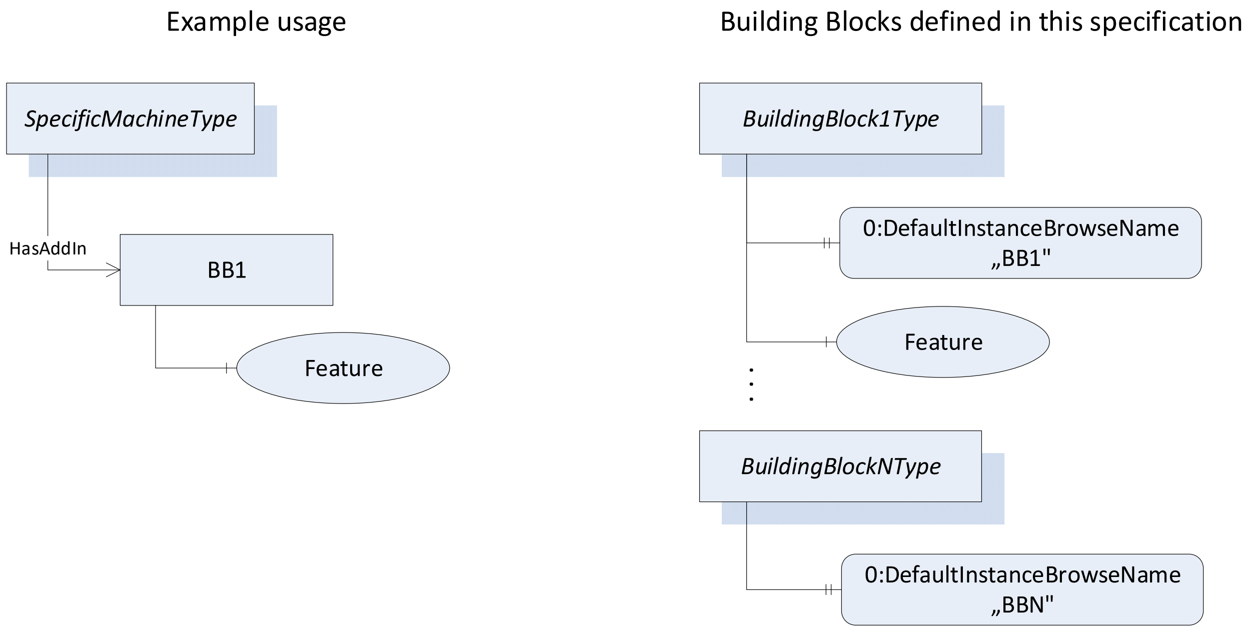
Example usage
Building Blocks defined in this specification
SpecificMachineType
HasAddIn
BB1
Feature
BuildingBlock1Type
0:DefaultInstanceBrowseName „BB1"
Feature
BuildingBlockNType
0:DefaultInstanceBrowseName „BBN"

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

7 General Recommendations

7.1 Localization

If the content of a *LocalizedText*, like the *Manufacturer* or the *Model* of a *Machine*, is language neutral, i.e. it is the same in all languages, the LocaleId 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 writeable, 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 7 gives an overview. The *AddIn MachineIdentificationType* with the default name "2:Identification" (as defined in OPC UA Part 100), is derived from the *2:FunctionalGroupType* and implements the interfaces *IMachineVendorNameplateType* and *IMachineTagNameplateType*. *IMachineVendorNameplateType* is a subtype of the *2:IVendorNameplateType* defined in OPC UA Part 100 and refines the usage of the *Properties* defined in that Interface, changes some to *Mandatory* and defines additional *Properties*. *IMachineTagNameplateType* is a subtype of the *2:ITagNameplateType* defined in OPC UA Part 100 and refines the usage of the *Properties* defined in that interface, and defines an additional *Property*.

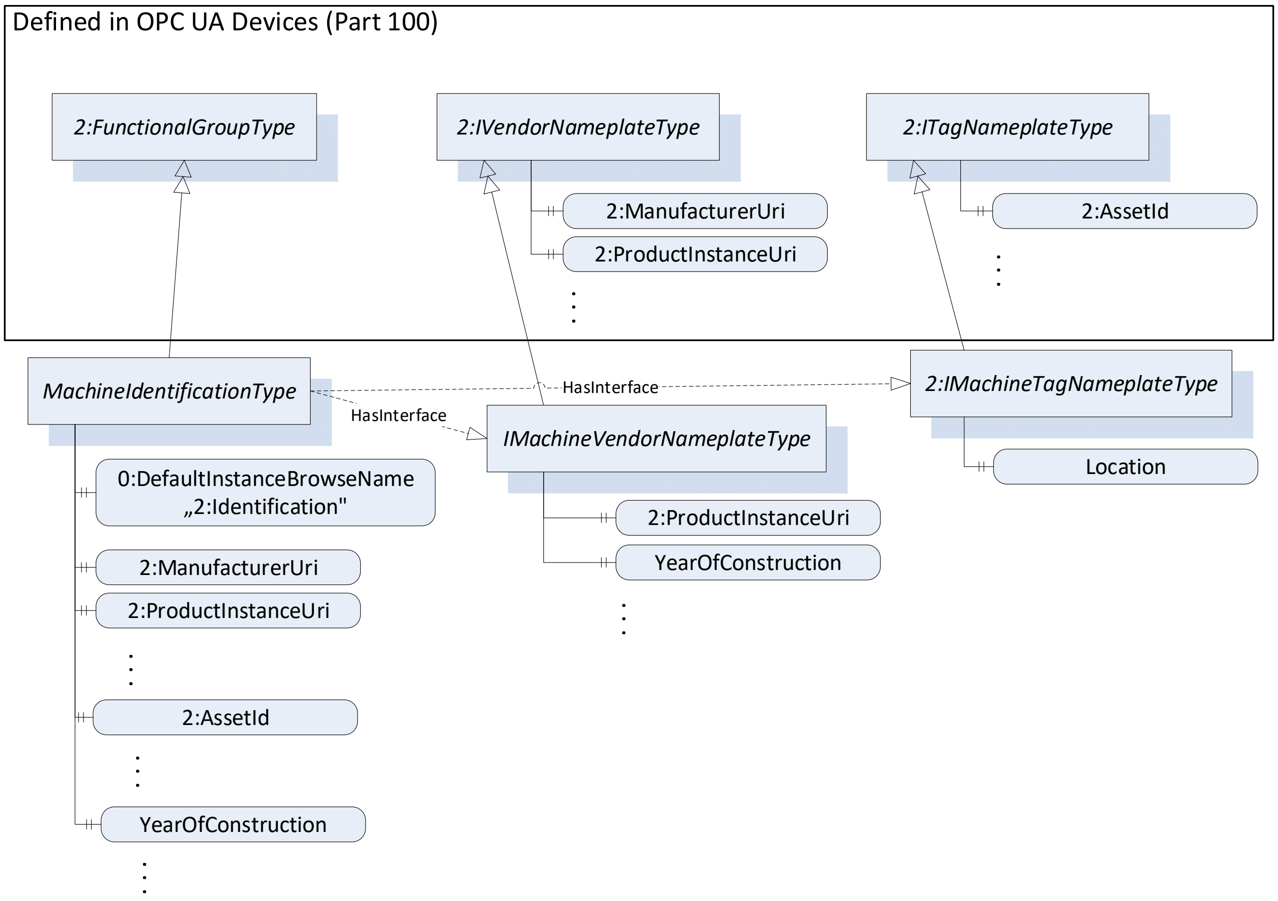

Figure 7 – Building Block for Identification and Nameplate

8.2 IMachineVendorNameplateType

The *IMachineVendorNameplateType* is a subtype of the *2:IVendorNameplateType* defined in Part 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 9.

Table 9 – 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 2:IVendorNameplateType defined in Part 100, i.e. inheriting the InstanceDeclarations of that Node.					
Properties of the IvendorNameplateType					
0:HasProperty	Variable	2:ProductInstanceUri	0:String	0:PropertyType	M, RO
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	Uint16	0:PropertyType	O, RO
0:HasProperty	Variable	MonthOfConstruction	Byte	0:PropertyType	O, RO
0:HasProperty	Variable	InitialOperationDate	DateTime	0:PropertyType	O, RO

The mandatory *2:ProductInstanceUri* is a globally unique resource identifier provided by the manufacturer of the *Machine*. It is defined by the *2:IvendorNameplateType*. It is intended to uniquely identify the *Machine* and shall not change during the life-cycle of the *Machine*. The length is restricted to 255 characters and it is the responsibility of the manufacturer that the *2:ProductInstanceUri* is globally unique. The syntax of the *2:ProductInstanceUri* is: <ManufacturerUri>/<any string>. The manufacturer might choose the serial number of the *Machine* as <any string>, if the serial number is unique within the manufacturer's scope, or a combination of machine model and serial number, if the serial number is only unique within a machine 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 *Machine*. 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 shall be used as Prefix in the *2:ProductInstanceUri*. As *2:ManufacturerUri* is used inside the *2:ProductInstanceUri*, it shall not change during the life-cycle of the *Machine*, 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 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 *Machine*, 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 optional 2:*Model* provides a human-readable, localized name of the model of the *Machine*. 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 *Machine* 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 machine, that might include options like the hardware configuration of the model. This information might be provided by the ERP system of the machine vendor and can be used for example 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 *Machine*. Examples are "11182372", "2377636".

The mandatory *2:SerialNumber* is a string containing a unique production number of the manufacturer of the machine. 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 *Machine*. 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 *2:HardwareRevision* provides a string representation of the revision level of the hardware of a machine. 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 *Machine*. Examples are: “01.33”, “A2”, “014/15120129-2018”.

The optional 2:*SoftwareRevision* provides a string representation of the revision level of a *Machine*. It is defined by the *2:IvendorNameplateType*. In most cases, *Machines* 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 *Machine*. Examples are: “PLL01 1.10.0.3” “V05.01.01.15”, “3.1 R1293”, “70.0.1”, “4.60.03”.

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

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

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

The optional *MonthOfConstruction* provides the month in which the manufacturing process of the *Machine* 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 *Machine* was switched on the first time after it has left the manufacturer plant. For example, “2020-01-29T18:59:59Z”, “2022-11-1712:00:00Z”.

The *InstanceDeclarations* of the *IMachineVendorNameplateType* have additional *Attributes* defined in Table 10.

Table 10 – IMachineVendorNameplateType Additional Attributes

Source Path	**Value**	**Description**
2:ProductInstanceUri	-	A globally unique resource identifier provided by the manufacturer of the machine
2:Manufacturer	-	A human-readable, localized name of the manufacturer of the machine
2:SerialNumber	-	A string containing a unique production number of the manufacturer of the machine. 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 machine.
YearOfConstruction	-	The year (Gregorian calendar) in which the manufacturing process of the machine has been completed. It shall be a four-digit number and never change during the life-cycle of a machine.
MonthOfConstruction	-	The month in which the manufacturing process of the machine has been completed. It shall be a number between 1 and 12, representing the month from January to December.
InitialOperationDate	-	The date, when the machine was switched on the first time after it has left the manufacturer plant.

8.3 IMachineTagNameplateType

The *ImachineTagNameplateType* is a subtype of the *2:ItagNameplateType* defined in Part 100. It refines the semantics of the *Properties* defined in *2:ItagNameplateType*, and adds an additional *Property*. It is formally defined in Table 11.

Table 11 – 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 Part 100, i.e. inheriting the InstanceDeclarations of that Node.					
0:HasProperty	Variable	Location	0:String	0:PropertyType	O, RW

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. The minimum number of Unicode characters of this *Property* shall be 40 characters, i.e. 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. The minimum number of Unicode characters for the text part of each locale shall be 40 characters, i.e. clients can expect to be able to write strings 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. The minimum number of Unicode characters of this *Property* shall be 60 characters, i.e. clients can expect to be able to write strings with a length of 40 Unicode characters into that field. Examples are “Munich/A2/217”, “Area 51”.

The *InstanceDeclarations* of the *IMachineTagNameplateType* have additional *Attributes* defined in Table 12.

Table 12 – IMachineTagNameplateType Additional Attributes

Source Path	**Value**	**Description**
Location	-	To be used by end users to store the location of the machine in a scheme specific to the end user. The minimum number of Unicode characters of this Property shall be 60 characters, i.e. clients can expect to be able to write strings with a length of 40 Unicode characters into that field.

8.4 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 13.

Table 13 – 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 2:FunctionalGroupType defined in Part 100, 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
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	UInt16	0:PropertyType	O, RO
0:HasProperty	Variable	MonthOfConstruction	Byte	0:PropertyType	O, RO
0:HasProperty	Variable	InitialOperationDate	DateTime	0:PropertyType	O, RO
Applied from IMachineTagNameplateType					
0:HasProperty	Variable	2:AssetId	0:String	0:PropertyType	O, RW
0:HasProperty	Variable	2:ComponentName	0:LocalizedText	0:PropertyType	O, RW
0:HasProperty	Variable	Location	0:String	0:PropertyType	O, RW

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

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.

The *InstanceDeclarations* of the *MachineIdentificationType* have additional *Attributes* defined in Table 14.

Table 14 – MachineIdentificationType Additional Attributes

Source Path	Value	Description
DefaultInstanceBrowseName	{"namespaceIndex":2, "name":Idenfification}	The default BrowseName for instances of the type.
2:ProductInstanceUri	-	A globally unique resource identifier provided by the manufacturer of the machine
2:Manufacturer	-	A human-readable, localized name of the manufacturer of the machine
2:ManufacturerUri	-	A globally unique identifier of the manufacturer of the machine
2:Model	-	A human-readable, localized name of the model of the machine
2:ProductCode	-	A machine-readable string of the model of the machine, that might include options like the hardware configuration of the model. This information might be provided by the ERP system of the machine vendor and can be used for example as order information.
2:HardwareRevision	-	A string representation of the revision level of the hardware of a machine. 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 machine. In most cases, machines 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 machine.
2:DeviceClass	-	Indicates in which domain or for what purpose the machine is used.
2:SerialNumber	-	A string containing a unique production number of the manufacturer of the machine. 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 machine.
YearOfConstruction	-	The year (Gregorian calendar) in which the manufacturing process of the machine has been completed. It shall be a four-digit number and never change during the life-cycle of a machine.
MonthOfConstruction	-	The month in which the manufacturing process of the machine has been completed. It shall be a number between 1 and 12, representing the month from January to December.
InitialOperationDate	-	The date, when the machine 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. The minimum number of Unicode characters of this Property shall be 40 characters, i.e. 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 machine. The minimum number of locales supported for this Property shall be two. The minimum number of Unicode characters for the text part of each locale shall be 40 characters, i.e. clients can expect to be able to write strings with a length of 40 Unicode characters into that field.
Location	-	To be used by end users to store the location of the machine in a scheme specific to the end user. The minimum number of Unicode characters of this Property shall be 60 characters, i.e. clients can expect to be able to write strings with a length of 40 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 8 gives an overview. There is a well-defined *Object* in the *AddressSpace* as entry point to browse to *Objects* representing a machine.

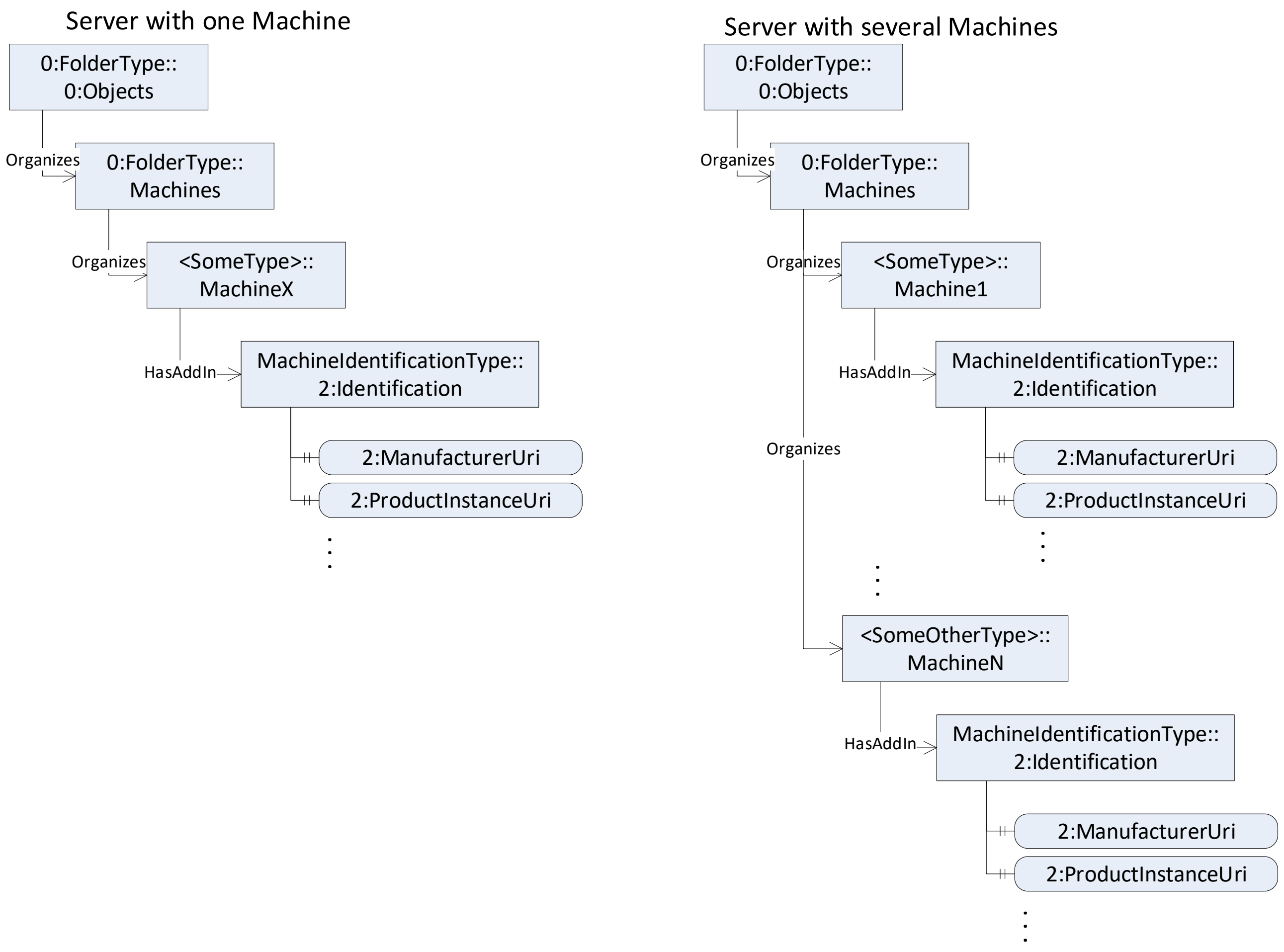

Figure 8 – Building Block for Finding all Machines in Server

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 15. 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 15 – 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 Part 5				
0:HasTypeDefinition	ObjectType	0:FolderType	Defined in Part 5	

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

The *Machines Object* shall be referenced from the *0:Objects Object* defined in OPC UA Part 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 Profiles and ConformanceUnits

10.1 Conformance Units

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

Table 16 – Conformance Units for OPC UA for Machinery

Category	Title	Description
Server	Machinery Identification	Supports the MachineIdentificationType with all its mandatory InstanceDeclarations, and optionally the optional InstanceDeclarations with read access.
Server	Machinery 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.
Server	Find Machines	Supports the Machines Object and references all Machines of the Server as defined by the Machines Object.

10.2 Profiles

10.2.1 Profile list

Table 17 lists all *Profiles* defined in this document and defines their URIs.

Table 17 – Profile URIs for OPC UA for Machinery

Profile	URI
Machinery Identification Server Facet	http://opcfoundation.org/UA-Profile/Machinery/Server/Identification
Machinery Identification Writable Server Facet	http://opcfoundation.org/UA-Profile/Machinery/Server/IdentificationWritable

10.2.2 Server Facets

10.2.2.1 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*.

10.2.2.2 Machinery Identification Server Facet

Table 18 defines a *Profile* that provides the identification of machines managed in an OPC UA Server.

Table 18 – Machinery Identification Server Facet

Group	Conformance Unit / Profile Title	M / O
Address Space Model	Address Space Base	M
Address Space Model	Address Space Interfaces	M
Address Space Model	Address Space AddIn Reference	M
Address Space Model	Address Space AddIn DefaultInstanceBrowsename	M
View Services	View Basic	M
View Services	View TranslateBrowsePath	M
View Services	View Minimum Continuation Point 01	M
Attribute Services	Attribute Read	M
Machinery	Machinery Identification	M
Machinery	Find Machines	M

10.2.2.3 Machinery Identification Writable Server Facet

Table 19 defines a *Facet* that provides the identification of machines as well as the writing of machine identification aspects changable by the user via the OPC UA interface.

Table 19 – Machinery Identification Writable Server Facet

Group	Conformance Unit / Profile Title	M / O
Profile	Machinery Identification Server Facet	
Attribute Services	Attribute Write Values	M
Machinery	Machinery Identification Writable	M

10.2.3 Client Facets

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

11 Namespaces

11.1 Namespace Metadata

Table 20 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 20 – NamespaceMetadata Object for this Document

Attribute	Value		
BrowseName	http://opcfoundation.org/UA/Machinery/		
References	**BrowseName**	**DataType**	**Value**
HasProperty	NamespaceUri	String	http://opcfoundation.org/UA/Machinery/
HasProperty	NamespaceVersion	String	1.0
HasProperty	NamespacePublicationDate	DateTime	2020-06-01
HasProperty	IsNamespaceSubset	Boolean	Vendor-specific
HasProperty	StaticNodeIdTypes	IdType []	Null
HasProperty	StaticNumericNodeIdRange	NumericRange []	Null
HasProperty	StaticStringNodeIdPattern	String	Null

11.2 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 21 provides a list of mandatory and optional namespaces used in an OPC UA for Machinery OPC UA *Server*.

Table 21 – 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 22 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 22 – 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 *InititalOperationDate 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/1.0/Opc.Ua.Machinery.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/Opc.Ua.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/1.0/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 B.1, 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.

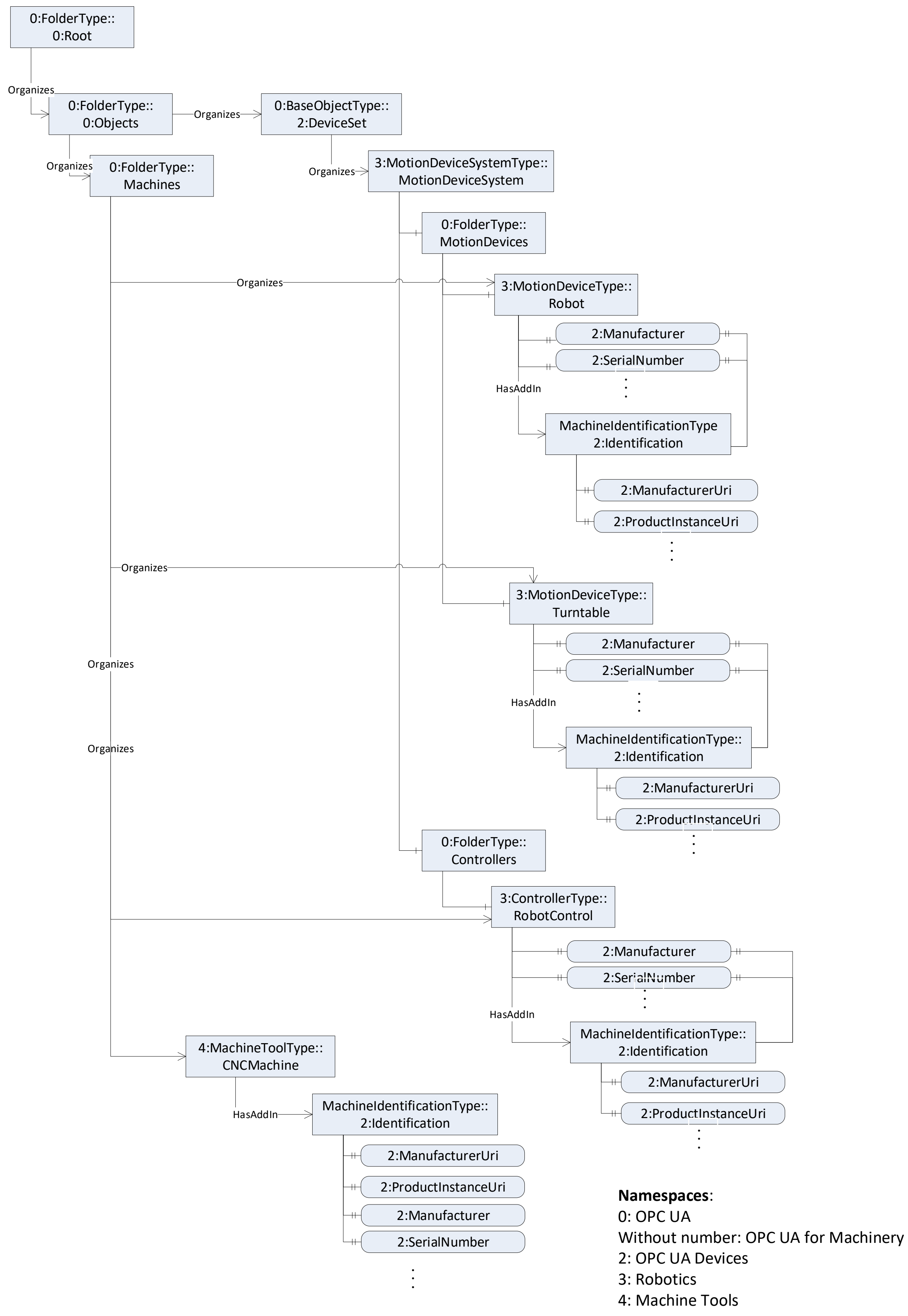

Figure B.1 – Example of Identification

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, *OPC for Machine Tools and Manufacturing Systems*
(http://www.opcfoundation.org/UA/MachineTools/)

[4] OPC 40010-1, *OPC UA for Robotics - Part 1: Vertical Integration*
(http://www.opcfoundation.org/UA/Robotics/)