Logo of OPC Foundation, only if this is a joint work. Both logos shall be side by side.

Logo of other organization. Both logos shall be side by side.

OPC nnnnn-m

OPC UA Companion Specification

OPC UA for <Title>

Part <mm>: <Part Name>

Draft 1.xy

YYYY-MM-DD

Edit the advanced properties to complete the title page.

Boxes are used to provide context or guidelines on expected content. Any words or phrases written in red font colour shall be replaced by actual values as appropriate for the companion specification.

Specification Type:	Industry Standard Specification	Comments:	
Document Number	**OPC nnnnn-m**		
Title:	OPC UA for <Title> Part <mm> :<Part Name>	Date:	YYYY-MM-DD
Version:	Draft 1.xy	Software:	MS-Word
		Source:	OPC 11020 - UA Companion Specification Template RC 1.01.11.docx
Author:	<organization>	Status:	Draft

Template Revisions

Version	Date	Description
1.01.00	Sept 02, 2019	New title page style that includes document numbers. Updates to normative references. Separated Profiles and Namespaces into two chapters and restructured the profile section. Added new tables for complex DataTypes, ReferenceTypes, and Instances. Added table for "additional references" that cannot be defined in the base type table. Clarified the use of namespace indexes for BrowseNames.
1.01.03	Oct 18, 2019	Clarifications and language improvements for description of Namespaces (4.2.3.2). Added a revision table (before clause 1 "Scope") that should be used in companion specification to list the changes to the previous revision.
1.01.04	Oct 28, 2019	Added a new example and the required tables to define methods. M-5209 Added custom document property "TemplateVersion". M-5316
1.01.05	Dec 20, 2019	Added the required tables to define union datatypes (M-5329), subcomponents (M-5315), and additional Variable attributes (M-5314 and M-5330)
1.01.07	Jan 14, 2019	Review meeting with additional editorial updates.
1.01.08	Feb 27, 2020	M-5386: revised table format for enumerations
1.01.09	Mar 09, 2020	M-4665: Link to NodeSet indicates that also ERRATA, Amendments or Revisions are applied. Fixed various styles.
1.01.10	Apr 29, 2020	Minor update for the Value attribute (M-5314).
1.01.11	July 09, 2020	M-5595: Added better wording and recommendations for NamespaceMetadata as suggested. M-5520: Fixed quotation marks. M-5781: Title "Additional Variable Attributes" misleading. M-5673: Added HasInterface Reference to figure. M-5468: Use of Namespace Index also in Method signatures. M-5789: Clarify use of NamespaceIndex for Profiles or ConformanceUnits in other specifications.

CONTENTS

1 Scope 8
2 Normative references 8
3 Terms, abbreviated terms and conventions 10
3.1 Overview 10
3.2 OPC UA for <title> terms 10
3.3 Abbreviated terms 10
3.4 Conventions used in this document 10
3.4.1 Conventions for Node descriptions 10
3.4.2 NodeIds and BrowseNames 12
3.4.3 Common Attributes 13
4 General information to <title> and OPC UA 15
4.1 Introduction to <title> 15
4.2 Introduction to OPC Unified Architecture 15
4.2.1 What is OPC UA? 15
4.2.2 Basics of OPC UA 15
4.2.3 Information modelling in OPC UA 16
5 Use cases 20
6 <title> Information Model overview 21
7 OPC UA ObjectTypes 21
7.1 <some>Type ObjectType definition 21
7.1.1 Overview 21
7.1.2 <some>Method 24
8 OPC UA EventTypes 25
8.1 <some>EventType 25
9 OPC UA VariableTypes 26
9.1 <some>VariableType 26
10 OPC UA DataTypes 26
10.1 <someStructure> 26
10.2 <someUnion> 26
10.3 <someEnumeration> 27
10.4 <someOptionSet> 27
11 OPC UA ReferenceTypes 28
11.1 <someReferenceType> 28
12 Instances 28
12.1 <someInstance> 28
13 Profiles and ConformanceUnits 29
13.1 Conformance Units 29
13.2 Profiles 29
13.2.1 Profile list 29
13.2.2 Server Facets 30
13.2.3 Client Facets 31
14 Namespaces 31
14.1 Namespace Metadata 31

14.2 Handling of OPC UA Namespaces 32
Annex A (normative) <Title> Namespace and mappings 34
A.1 Namespace and identifiers for <Title> Information Model 34

FIGURES

Figure 1 – The Scope of OPC UA within an Enterprise 16
Figure 2 – A Basic Object in an OPC UA Address Space 17
Figure 3 – The Relationship between Type Definitions and Instances 18
Figure 4 – Examples of References between Objects 19
Figure 5 – The OPC UA Information Model Notation 19

TABLES

Table 1 – Examples of DataTypes 11
Table 2 – Type Definition Table 12
Table 3 – Examples of Other Characteristics 12
Table 4 – Common Node Attributes 13
Table 5 – Common Object Attributes 13
Table 6 – Common Variable Attributes 14
Table 7 – Common VariableType Attributes 14
Table 8 – Common Method Attributes 14
Table 9 – <some>Type Definition 21
Table 10 – <some>Type Additional References 21
Table 11 – <some>Type Additional Subcomponents 23
Table 12 – <some>Type Attribute values for child Nodes 24
Table 13 – <some>Method Method Arguments 25
Table 14 – <some>Method Method AddressSpace definition 25
Table 15 – <some>EventType Definition 25
Table 16 – <some>Type Definition 26
Table 17 – <someStructure> Structure 26
Table 18 – <someStructure> Definition 26
Table 19 – <someUnion> Union 26
Table 20 – <someUnion> Definition 27
Table 21 – <someEnumeration> Items 27
Table 22 – <someEnumeration> Definition 27
Table 23 – <someOptionSet> Values 27
Table 24 – <someOptionSet> Definition 28
Table 25 – <someReferenceType> Definition 28
Table 27 – <someInstance> Definition 28
Table 28 – Conformance Units for <Title> 29
Table 29 – Profile URIs for <Title> 29
Table 30 - <short name> <Prf1name> Server Profile 30
Table 31 - <short name> <Prf2name> Server Facet 31
Table 32 - <short name> < Prf3name> Client Facet 31
Table 33 – NamespaceMetadata Object for this Document 32
Table 34 – Namespaces used in a <title> Server 32
Table 35 – Namespaces used in this document 33

<OPC FOUNDATION (IF JOINT WORK)>, <OTHER ORGANIZATION>

<Remove the above Public Domain notice and add a copyright date in the following Agreement of use for the completed specification. If possible (but not legally required), provide attribution to the original author in the following way:

"This specification is a derivative work of the public domain document OPC UA Companion Specification Template, by OPC Foundation"
>

AGREEMENT OF USE

<For joint work, this is the agreement accepted by OPC Foundation and used by most if not all joint companion specifications.
Changes to this agreement will require OPC Foundation review. For non-joint work replace this section with appropriate agreement of use and copyright verbiage.>

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 <other organization> 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 <other organization> 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.

Revision x.y Highlights

<Revision Highlights specify the interesting changes to the previous revision. Such a "Revision Highlights" clause will not exist in the first revision.>

The table below follows the design and process used for the OPC UA specification. All "interesting" changes need to be reported in the problem tracking tool used for the specification. The OPC Foundation uses Mantis and provides it for companion specs as well. If a different problem tracking tool is used, "Mantis" needs to be replaced.

<The two rows in the following table are just examples. Note that the "Mantis IDs" are real hyperlinks – when clicked the reader opens the Mantis page.>

The following table includes the Mantis issues resolved with this revision.

Mantis ID	Summary	Resolution
3165	Wrong variable data types in Program example.	Fixed datatypes and replaced Table A.14 by description of the variables.
3521	Modelling rules for states and transitions.	Removed the modelling rules from state and transition objects as they are never created (multiple tables).

MAIN TITLE IN CAPITAL LETTERS –

Part X: Second part of the title in normal letters

1 Scope

This document XXXXX specifies / establishes / …

<Specify what this document covers. Look into other companion specs for examples.>

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

<other organization>

2 Normative references

The following referenced documents are indispensable for the application 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.

For references to the OPC UA Specification it is recommended to define the minimum required version. Example: “1.04.03 is the minimum required version for the following OPC Unified Architecture parts.”
The build number of the version (in this case “03”) refers to an ERRATA document with the corresponding version (see https://opcfoundation.org/developer-tools/specifications-unified-architecture/errata-and-amendments/ for the published ERRATA documents).

<Insert only references that apply to this document. Following are examples only>

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 10000-8, *OPC Unified Architecture - Part 8: Data Access*
http://www.opcfoundation.org/UA/Part8/

OPC 10000-9, *OPC Unified Architecture - Part 9: Alarms and Conditions*
http://www.opcfoundation.org/UA/Part9/

OPC 10000-10, *OPC Unified Architecture - Part 10: Programs*
http://www.opcfoundation.org/UA/Part10/

OPC 10000-11, *OPC Unified Architecture - Part 11: Historical Access*
http://www.opcfoundation.org/UA/Part11/

OPC 10000-12, *OPC Unified Architecture - Part 12: Discovery and Global Services*
http://www.opcfoundation.org/UA/Part12/

OPC 10000-13, *OPC Unified Architecture - Part 13: Aggregates*
http://www.opcfoundation.org/UA/Part13/

OPC 10000-14, *OPC Unified Architecture - Part 14: PubSub*
http://www.opcfoundation.org/UA/Part14/

OPC 10001-1, *OPC Unified Architecture V1.04 - Amendment 1: AnalogItem Types*
http://www.opcfoundation.org/UA/Amendment1/

OPC 10001-3, *OPC Unified Architecture V1.04 - Amendment 3: Method Metadata*
http://www.opcfoundation.org/UA/Amendment3/

OPC 10001-5, *OPC Unified Architecture V1.04 - Amendment 5: Dictionary Reference*
http://www.opcfoundation.org/UA/Amendment5/

OPC 10001-7, *OPC Unified Architecture V1.04 - Amendment 7: Interfaces ad AddIns*
http://www.opcfoundation.org/UA/Amendment7/

OPC 10001-11, *OPC Unified Architecture V1.04 - Amendment 11: Spatial Types*

http://www.opcfoundation.org/UA/Amendment11/

OPC 10000-100, *OPC Unified Architecture - Part 100: Devices*
http://www.opcfoundation.org/UA/Part100/

3 Terms, abbreviated terms and conventions

3.1 Overview

It is assumed that basic concepts of OPC UA information modelling and <other specifications> are understood in this document. This document will use these concepts to describe the <title> 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, OPC 10000-100, ... as well as the following apply.

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

3.2 OPC UA for <title> terms

3.2.1
term 1
<a short description – max two lines>

Note 1 to entry: Optional additional text if the short description is not considered sufficient.

EXAMPLE 1 First example for term 1.

EXAMPLE 2 Second example for term 1.

[SOURCE: where definition 1 was found]

3.2.2
term 2
definition 2

3.3 Abbreviated terms

The following abbreviations are examples. The list shall only contain abbreviations used in the document.

AC	Alarm and Condition
DCS	Distributed Control Systems

3.4 Conventions used in this document

Following are basic conventions that shall be followed for all formal definitions used.

3.4.1 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	ArrayDimensions	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 Other 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 "--" is used.				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
ReferenceType name	*NodeClass* of the *TargetNode*.	*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.					

批注 **[RSA1]:** This column was made more generic in the 1.01 version of the template.

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 3.4.3.1. 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.

3.4.2 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.

If the *BrowseName* is not defined by this document, a namespace index prefix is added to the *BrowseName* (e.g., prefix '0' leading to '0:EngineeringUnits' or prefix '2' leading to '2:DeviceRevision'). This is typically necessary if a *Property* of another specification is overwritten or used in the OPC UA types defined in this document. Table 34 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 document or if it is server-specific.

For all *Nodes* specified in this document, 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* are 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 document.
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 depends 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 document, the *Attributes* named in Table 5 shall be set as specified in the Table 5. 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.

3.4.3.3 Variables

For all *Variables* specified in this document, 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 document, 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 document, 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*.

3.4.3.5 Methods

For all *Methods* specified in this document, 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 document 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 <title> and OPC UA

4.1 Introduction to <title>

Insert an introduction (about one page) of the companion organization and the model that it represents.

4.2 Introduction to OPC Unified Architecture

This is an OPC UA introduction that may be used as is, shortened or enhanced as appropriate.

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 <title>, OPC UA makes it easier for end users to access data via generic commercial applications.

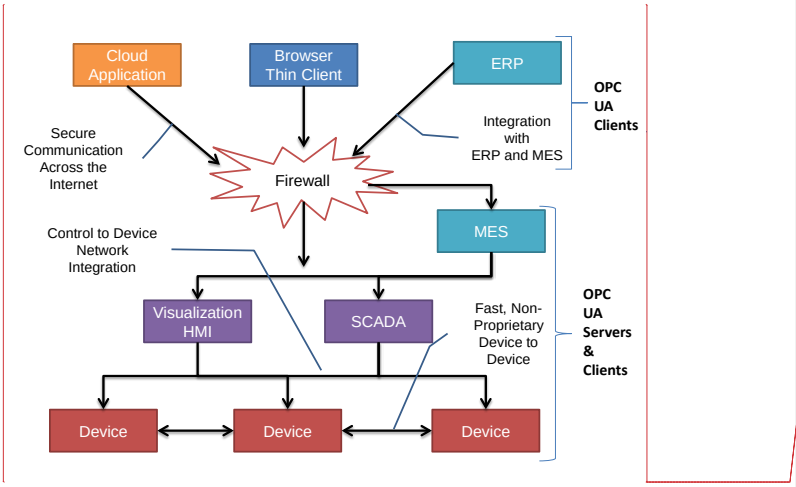
The OPC UA model is scalable from small devices to ERP systems. OPC UA *Servers* process information locally and then provide that data in a consistent format to any application requesting data - ERP, MES, PMS, Maintenance Systems, HMI, Smartphone or a standard Browser, for examples. For a more complete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



批注 **[RSA2]:** This figure is an embedded PowerPoint slide object. Figures shall always be embedded objects or simple images. They shall not be diagrams created within Word.

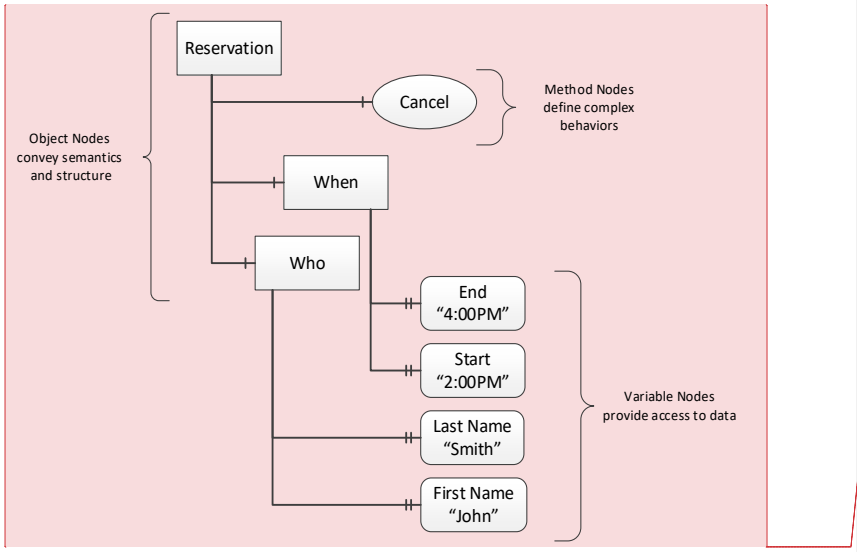
Figure 1 – The Scope of OPC UA within an Enterprise

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.



批注 **[RSA3]:** This figure is an embedded Visio object.

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 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.

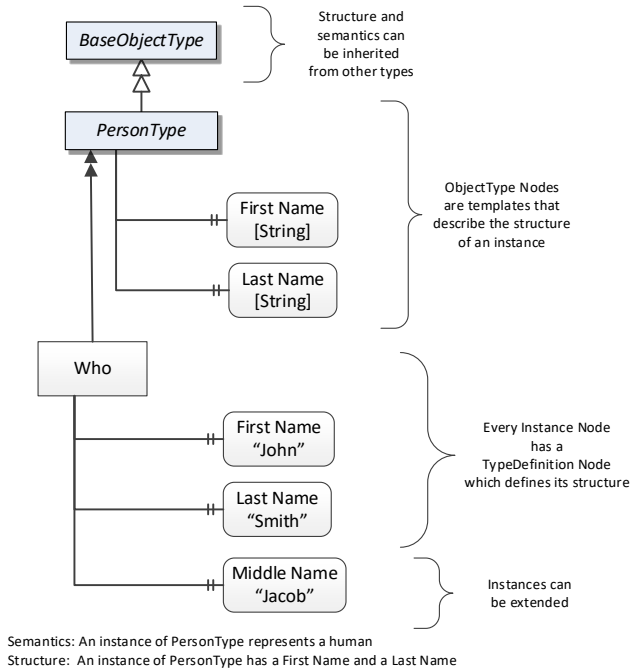


Figure 3 – The Relationship between Type Definitions and Instances

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*.

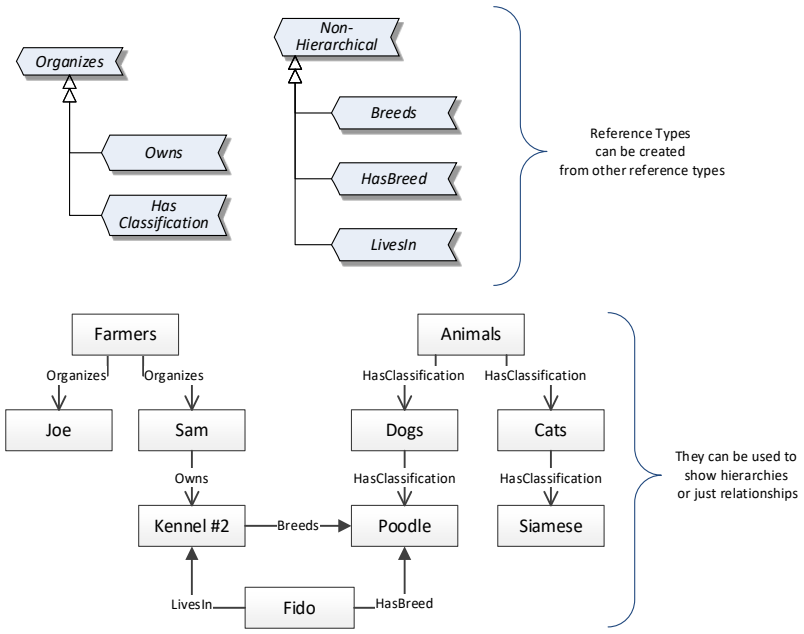


Figure 4 – Examples of References between Objects

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*.

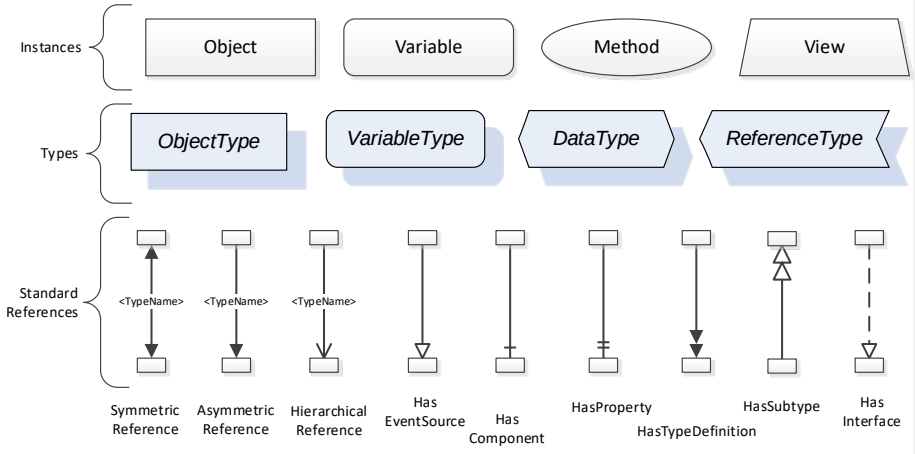


Figure 5 – The OPC UA Information Model Notation

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

Insert the use cases that can be achieved by using OPC UA with the companion organization's information model.

6 <title> Information Model overview

An overview of the model elements and how they relate to each other.

Following shall be sections that specify the companion information model. Such models may vary and no fixed structure can be given. An option could be to have separate chapters for ObjectTypes, VariableTypes, DataTypes, a.s.o.

7 OPC UA ObjectTypes

7.1 <some>Type ObjectType definition

7.1.1 Overview

The <some>*Type* provides ... and is formally defined in Table 9.

Table 9 – <some>Type Definition

Attribute	Value				
BrowseName	<some>Type				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the <other>Type defined in ..., i.e. inheriting the InstanceDeclarations of that Node.					
0:HasProperty	Variable	<some>Property1	0:String	0:PropertyType	M, RO
0:HasProperty	Variable	<some>Property2	0:Int32	0:PropertyType	M, RW
0:HasComponent	Variable	<some>Measurement	0:Double	0:AnalogItemType	O
0:HasComponent	Object	<some>Alarm		0:AlarmConditionType	O
0:HasComponent	Method	<some>Method	See 7.1.2		M
0:HasDictionaryEntry	Object	3:0112/2///61987#xzx607			

批注 **[KD(4]:** The abbreviations are described in Table 3.

The components of the <some>*Type* have additional references which are defined in Table 10.

Table 10 – <some>Type Additional References

Source Path	Reference Type	Is Forward	Target Path
<some>Property1	0:HasDictionaryEntry	True	3:0112/2///61987#xzx608
<some>Measurement 0:EngineeringUnits	0:Organizes	False	0:Objects Units
<some>Measurement	0:HasCondition	True	<some>Type <some>Alarm

批注 **[RSA5]:** The first element in the path refers to a type or well known instance which is uniquely identified within a namespace by the BrowseName.

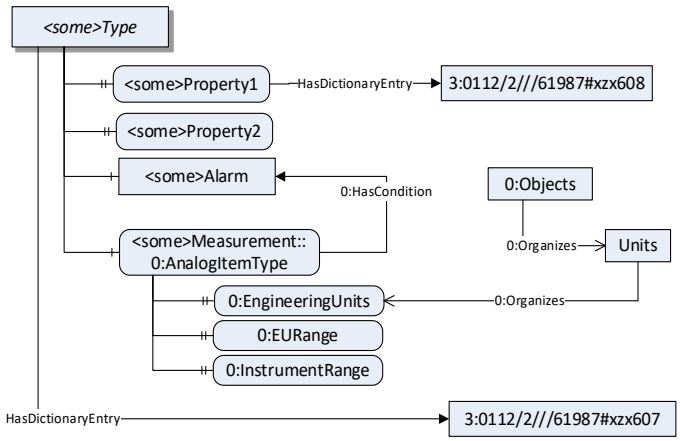
批注 **[RSA6]:** Multiple elements in the BrowsePath are added in separate rows of nested table. Note the spacer paragraphs before and after (with an additional space after).

SPECIFYING ADDITIONAL REFERENCES

Table 9 allows you to define *ObjectTypes*. You add *InstanceDeclarations*, which can be based on complex *TypeDefinitions* (such as *AnalogItemType*). The complex structure of those *TypeDefinitions* does not need to be further defined, as it is already done by their *TypeDefinitions*. However, if you want to add additional *References*, you can use a table format as shown in Table 10. This format allows you to add *References* from those *InstanceDeclarations* to any other *Node*. The IsForward column indicates whether to use a forward or inverse *Reference*. The Source Path is always relative to the *TypeDefinition*, and

can be a *BrowsePath* as in the second entry of the table. The Target Path points to another *Node*, which can be a well-known instance or a *TypeDefinition*. You can use *BrowsePaths* here as well, but the first entry needs to be the *BrowseName* of the starting *Node*.

In the following figure shows of the *AddressSpace* as defined by Table 9 and Table 10.



SPECIFYING DICTIONARY REFERENCES

When making use of dictionary references (see OPC 10001-5) the following rules apply:

- *DictionaryEntries* from a *TypeDefinition* shall be referenced directly (see Table 9)
- *DictionaryEntries* from *InstanceDeclarations* shall be referenced by using an 'Additional References' table for the *TypeDefinition* (see Table 10).
- When using IRDIs or URIs
  - use the well-defined namespace (see OPC 10001-5) and include this in Table 34. This template already contains as example the IRDI namespace
  - create an additional NodeSet file for this namespace containing only the *DictionaryEntries* your specification is referencing, no additional organizational structure
  - use the URI or IRDI as the *BrowseName* and add a prefixed for the *NamespaceIndex* (see examples above)

The components of the <some>*Type* have additional references which are defined in Table 11.

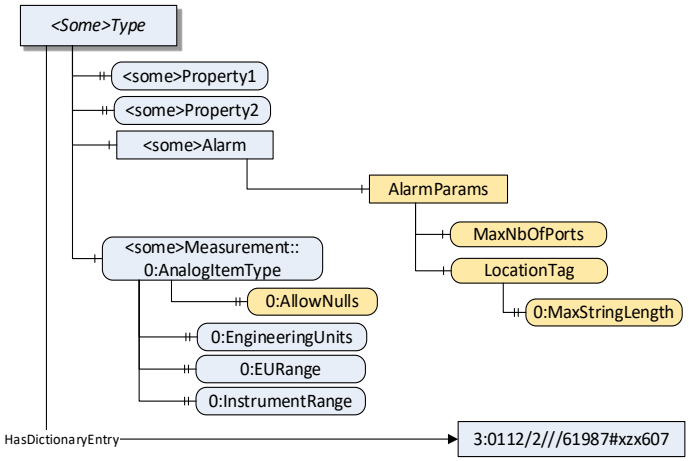
Table 11 – <some>Type Additional Subcomponents

Source Path	References	NodeClass	BrowseName	DataType	TypeDefinition	Others
<some>Alarm	0:HasComponent	Object	AlarmParams		0:BaseObjectType	M
<some>Measurement	0:HasProperty	Variable	0:AllowNulls	0:Boolean	0:PropertyType	O
<some>Alarm AlarmParams	0:HasComponent	Variable	MaxNumberOfPorts	0:Byte	0:BaseDataVariableType	M
<some>Alarm AlarmParams	0:HasComponent	Variable	LocationTag	0:String	0:BaseDataVariableType	M
<some>Alarm AlarmParams LocationTag	0:HasProperty	Variable	0:MaxStringLength	0:UInt32	0:PropertyType	O

SPECIFYING ADDITIONAL SUBCOMPONENTS

Table 9 allows you to define *ObjectTypes*. You add *InstanceDeclarations*, which can be based on complex *TypeDefinitions* (such as *AnalogItemType*). The complex structure of those *TypeDefinitions* does not need to be further defined, as it is already done by their *TypeDefinitions*. However, if you want to add additional Sub-components, you can use a table format as shown in Table 11. This format allows you to add instances to those *InstanceDeclarations*. The Source Path is always relative to the *TypeDefinition*.

In the following figure shows of the *AddressSpace* as defined by Table 9 and Table 11.



The component *Variables* of the <some>*Type* have additional *Attributes* defined in Table 12.

Table 12 – <some>Type Attribute values for child Nodes

Source Path	Value Attribute	Description Attribute
<some>Measurement	5.7	This is a description for <some>Measurement
<some>Alarm AlarmParams LocationTag	"Building 2"	This is a description for LocationTag.
<some>Measurement 0:EURange	High: 1000 Low: 0	This is the EURange for <some>Measurement.
<some>Measurement 0:EngineeringUnits	NamespaceUri: <some namespace> UnitId: 1234 DisplayName: Fidgets Description: <some description>	
<some>Alarm		This is a description for <some>Alarm

Fields may be empty which means this *Attribute* is not defined.

SPECIFYING ATTRIBUTE VALUES

The values of attributes are converted to text in the document 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. The double quotes are not included when entering JSON strings.

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.

7.1.2 <some>Method

Provide description of the method, what it is used for, how it works etc

If specific result codes are to be used, it is recommended to include the table "Method Result Codes" and include these specific codes.

The signature of this *Method* is specified below. Table 13 and Table 14 specify the *Arguments* and *AddressSpace* representation, respectively.

The *AddressSpace* definition can be omitted if there are no *Properties* other than *InputArguments* and *OutputArguments*.

Signature

```
<some>Method (
  [in]  0:String       InArg1,
  [in]  0:Float        InArg2,
  [out] 0:UInt32       OutArg1,
  [out] 0:Int32        someMethodStatus);
```

Table 13 – <some>Method Method Arguments

Argument	Description
InArg1	<description>
InArg2	<description>
OutArg1	<description>
someMethodStatus	This is an example where the Method needs to return special status information.
	0 – OK -1 – E_FirstError – <description> -2 – E_SecondError – <description>

Provide description of the method, what it is used for, how it works etc

Method Result Codes (defined in Call Service)

Result Code	Description
Bad_UserAccessDenied	See OPC 10000-4 for a general description.

Table 14 – <some>Method Method AddressSpace definition

Attribute	Value				
BrowseName	<some>Method				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
0:HasProperty	Variable	0:InputArguments	0:Argument[]	0:PropertyType	0:Mandatory
0:HasProperty	Variable	0:OutputArguments	0:Argument[]	0:PropertyType	0:Mandatory

8 OPC UA EventTypes

8.1 <some>EventType

This *EventType* is ….. Its representation in the *AddressSpace* is formally defined in Table 15.

Table 15 – <some>EventType Definition

Attribute		Value			
BrowseName		<some>EventType			
IsAbstract		True			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the *BaseEventType* defined in …, which means it inherits the InstanceDeclarations of that Node.					
0:HasSubtype	ObjectType	<someother>EventType	Defined in		
0:HasProperty	Variable	<some>Eventfield	0:String	0:PropertyType	0:Mandatory

This *EventType* inherits all *Properties* of the *BaseEventType*. ….

9 OPC UA VariableTypes

9.1 <some>VariableType

The <some>*VariableType* is a subtype of the *BaseVariableType*. It is used ….

It is formally defined in Table 16.

Table 16 – <some>Type Definition

Attribute		Value			
BrowseName		<some>Type			
IsAbstract		False			
ValueRank		−1 (−1 = Scalar)			
DataType		String			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the BaseDataVariableType defined in …					
0:HasComponent	Variable	<var1>	0:UtcTime	0:BaseDataVariableType	0:Mandatory
0:HasComponent	Variable	<var2>	0:UtcTime	0:BaseDataVariableType	0:Mandatory

10 OPC UA DataTypes

10.1 <someStructure>

This structure contains …. The structure is defined in Table 17.

Table 17 – <someStructure> Structure

Name	Type	Description
<someStructure>	structure	Subtype of <someParentStructure> defined in …
SP1	0:Byte[]	Setpoint 1
SP2	0:Byte[]	Setpoint 2

Its representation in the *AddressSpace* is defined in Table 18.

The *AddressSpace* definition can be omitted if isAbstract=false and there are no *Properties*.

Table 18 – <someStructure> Definition

Attribute		Value			
BrowseName		<someStructure>			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the <someParentStructure> defined in …					

10.2 <someUnion>

This union contains …. The union is defined in Table 19.

Table 19 – <someUnion> Union

Name	Type	Description
<someUnion>	union	
Var_1	0:String	First set
Var_2	<someStructure>	Second set
Var_3	<someEnumeration>	Third set

Its representation in the AddressSpace is defined in Table 20.

Table 20 – <someUnion> Definition

Attributes	Value
BrowseName	<someUnion>
IsAbstract	False
Subtype of Union defined in OPC 10000-5.	

10.3 <someEnumeration>

This enumeration .... The enumeration is defined in Table 21.

Table 21 – <someEnumeration> Items

Name	Value	Description
<Enum1_Name>	0	<Enum1Description>
<Enum2_Name>	1	<Enum2Description>
<Enum3_Name>	2	<Enum4Description>

Each *Enumeration* item is represented by a "Name" - the human readable representation and a "Value" - the numeric representation. If the *Enumeration* is zero-based and sequential, the *EnumStrings Property* is used for the names. In all other cases the *EnumValues Property* has to be used.

Its representation in the AddressSpace is defined in Table 22..

The *AddressSpace* definition can be omitted if isAbstract=false and there are no *Properties* other than *EnumStrings*.

Table 22 – <someEnumeration> Definition

Attribute		Value			
BrowseName		<someEnumeration>			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in OPC 10000-5					
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText []	0:PropertyType	

10.4 <someOptionSet>

This *DataType* defines flags for ... <someOptionSet> is formally defined in Table 23.

Table 23 – <someOptionSet> Values

Value	Bit No.	Description
<Value1>	0	This flag....
<Value2>	1	This flag....
<Value3>	2	This flag....

The <someOptionSet> representation in the *AddressSpace* is defined in Table 24.

Table 24 – <someOptionSet> Definition

Attribute		Value			
BrowseName		<someOptionSet>			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the OptionSet DataType defined in OPC 10000-5					
0:HasProperty	Variable	0:OptionSetValues	0:LocalizedText []	0:PropertyType	

11 OPC UA ReferenceTypes

11.1 <someReferenceType>

The <someReferenceType> is a concrete *ReferenceType* and can be used directly. It is a subtype of <someParentReferenceType>.

The semantic of this *ReferenceType* is to link ……

The *SourceNode* of *References* of this type shall be an……

The *TargetNode* of this *ReferenceType* shall be an …..

The <someReferenceType> is formally defined in Table 25.

Table 25 – <someReferenceType> Definition

Attributes	Value		
BrowseName	<someReferenceType>		
InverseName	<someinverseName>		
Symmetric	<True/False>		
IsAbstract	<True/False>		
References	**NodeClass**	**BrowseName**	**Comment**
Subtype <someParentReferenceType>			

12 Instances

12.1 <someInstance>

The <someInstance> is formally defined in Table 26.

Table 26 – <someInstance> Definition

Attribute	Value			
BrowseName	<someInstance>			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**
OrganizedBy by the <TheLocationInAddressSpace> defined in <Where It is Defined>				
0:HasTypeDefinition	<class of SomeInstance>	*<Type of someInstance>*	Defined in <Where Type of SomeInstance isdefined>	

Provide some description of the instance, what it is used for, constraints on it etc

13 Profiles and ConformanceUnits

Profiles and *ConformanceUnits* break functionality into testable groups. All companion specification shall include at least one *Profile*/*Facet*. If there are any groupings of functionality that not all *Servers*/*Client* would implement then multiple *Profile*/*Facet* are encouraged. A *ConformanceUnit* should describe a testable unit. A single *ConformanceUnit* is tested as a unit so all items covered by it must be support or the *ConformanceUnit* will fail. *ConformanceUnits* can be included in multiple *Profiles*, thus they are declared in their own table.

The name of the *Profile* should end with *Facet* or *Profile*. A *Facet* is a grouping of functionality that must also be paired with other *Facets* to create a running *Server* or *Client*. A *Profile* is all inclusive, in that is the *Profile* is implemented no additional functionality would be required to have a running application.

<short name>
A <short name> is required for each companion specification to assure uniqueness of string identifiers. It precedes the names of profiles and conformance units and is included in URIs and URLs defined in a companion specification.
A <short name> is all caps if an acronym, otherwise camel case.
Exception if the short name is a trademark. Use trademark casing.

13.1 Conformance Units

Table 27 defines the corresponding *ConformanceUnits* for the OPC UA Information Model for <title>.

Table 27 – Conformance Units for <Title>

Category	Title	Description
Server	<short name> <Function1>	Supports the base functionality defined in <Title> Information Model. This includes.....
Server	<short name> <Function2>	Supports the .....
Server	<short name> <Function3>	Supports the .....
Client	<short name> Client <Function1>	The client can make use of the .....

Typically, *Client ConformanceUnits* describe the use of a function, but they do not need to match 1 to 1 with *Server ConformanceUnits*. They might also reference to other categories defined in Part 7 (Pub, Sub, GDS...). For larger companion specifications, there might be separate tables for *Client ConformanceUnits*, *Server ConformanceUnits*, etc.

13.2 Profiles

13.2.1 Profile list

Table 28 lists all Profiles defined in this document and defines their URIs.

Table 28 – Profile URIs for <Title>

Profile	URI
<short name> <Prf1name> Server Profile	http://opcfoundation.org/UA-Profile/<short name>/Server/<Prf1name>
<short name> <Prf2name> Server Facet	http://opcfoundation.org/UA-Profile/<short name>/Server/<Prf2name>

Profile	URI
<short name> <Prf3name> Client Facet	http://opcfoundation.org/UA-Profile/<short name>/Client/<Prf3name>

13.2.2 Server Facets

13.2.2.1 Overview

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

A specification can define multiple *Facets* if not all features are to be implemented by all *Servers* and *Clients*. The name of the *Facet* shall give a hint of the subset. An overall description shall be provided that explains the subset and it potential use.

13.2.2.2 <short name> <Prf1name> Server Profile

Table 29 defines a *Profile* that describes the …….

Table 29 - <short name> <Prf1name> Server Profile

Group	Conformance Unit / Profile Title	Mandatory / Optional
Profile	0:Core 2017 Server Facet http://opcfoundation.org/UA-Profile/Server/Core2017Facet	
Profile	0:UA-TCP UA-SC UA Binary http://opcfoundation.org/UA-Profile/Transport/uatcp-uasc-uabinary	
Profile	0:Data Access Server Facet http://opcfoundation.org/UA-Profile/Server/DataAccess	
Profile	2:BaseDevice_Server_Facet	
Profile	<short name> <Prf2name> Server Facet	
Subscription Services	0:Subscription Durable	M
<short name>	<short name> <Function1>	M

This table lists a *Profile*, in which it includes other base *Profiles* that would be needed to make a working *Server*. It also includes other *Facets* defined in this companion specification and *ConformanceUnits* defined in this companion standard.

A namespace shall be included if Profiles or ConformanceUnits of another specification are included. In the example above '0' represents the OPC UA core specification and '2' UA for Devices (see Table 34).

The column with title "Mandatory / Optional" defines whether support of included *ConformanceUnits* is optional or mandatory. Optional means that an application has the option to not support the *ConformanceUnit*. However, if supported, the application shall pass all tests associated with the *ConformanceUnit*.

The "Group" for all Conformance Units defined in this document shall be the <short name>. If Conformance Units of OPC 10000-7 are referenced, the corresponding Groups shall be used. See the example with group "Subscription Services".

13.2.2.3 <short name><Prf2name> Server Facet

Table 30 defines a *Facet* that describes the …….

Table 30 - <short name> <Prf2name> Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
<short name>	<short name> <Function1>	M
<short name>	<short name> <Function3>	O

This table lists a *Facet*, in that it must be include with other *Facets* to create a running application. It defines the *ConformanceUnits* and other facets that are required

13.2.3 Client Facets

13.2.3.1 Overview

The following tables specify the *Facets* available for *Clients* that implement the <title> companion specification.

A specification can define multiple facets if not all features are to be implemented by all *Servers* and *Clients*. The name of the facet shall give a hint of the subset. An overall description shall be provided that explains the subset and it potential use.

13.2.3.2 <short name> < Prf3name> Client Facet

Table 31 defines a *Facet* that describes the base characteristics for all OPC UA *Clients* that make use of this companion specification. Additional *Profiles* will define support for various information models that are part of this document.

Table 31 - <short name> < Prf3name> Client Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Profile	0:AddressSpace Lookup Client Facet http://opcfoundation.org/UA-Profile/Client/AddressSpaceLookup	
Profile	0:DataAccess Client Facet http://opcfoundation.org/UA-Profile/Client/DataAccess	
Profile	0:DataChange Subscriber Client Facet http://opcfoundation.org/UA-Profile/Client/DataChangeSubscriber	
Session Services	0:Session Client Detect Shutdown	M
<short name>	<short name> Client <Function1>	M

This table lists a *Facet*, in that it must be include with other *Facets* to create a running application. It defines the *ConformanceUnits* and other facets that are required as an example it include other base Facets and a Base system *ConformanceUnit*

14 Namespaces

14.1 Namespace Metadata

Namespace Metadata are required for any companion standard that specifies an information model (e.g. *Objects* and *ObjectTypes*). The metadata provide standardized information about

the elements of this namespace. This information is particularly important for aggregating *Servers*.
Typically, all Nodes of a companion specification are static and therefore the metadata shall describe them as static. This is done by setting all Numeric NodeIds to static (StaticNodeIdTypes). If you use different NodeIds (e.g. Strings), this needs to be adapted. If not all Nodes are static, it needs to be adapted as well. Static NodeIds mean, that the same Node is used in all servers, e.g. for TypeDefinitions or entry points like the "Root" Object of the base specification. Not static Nodes would be Nodes providing server-specific information (e.g. typically all the instances based on the TypeDefinitions of a companion specification) or other dynamic behaviour (e.g. a standardized Method that adds or removes something from a server).

Table 32 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 32 – NamespaceMetadata Object for this Document

Attribute	**Value**		
BrowseName	http://opcfoundation.org/UA/<short name>/		
References	**BrowseName**	**DataType**	**Value**
HasProperty	NamespaceUri	String	http://opcfoundation.org/UA/<short name>
HasProperty	NamespaceVersion	String	X.YY
HasProperty	NamespacePublicationDate	DateTime	YYYY-MM-DD
HasProperty	IsNamespaceSubset	Boolean	True or False
HasProperty	StaticNodeIdTypes	IdType []	{Numeric}
HasProperty	StaticNumericNodeIdRange	NumericRange []	Null
HasProperty	StaticStringNodeIdPattern	String	Null

14.2 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 33 provides a list of mandatory and optional namespaces used in an <title> OPC UA *Server*.

Table 33 – Namespaces used in a <title> 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

NamespaceURI	Description	Use
Local Server URI	Namespace for nodes defined in the local server. This may include types and instances used in an AutoID Device represented by the 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/<title>/	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

批注 [DK7]:
This row is an example where an additional companion standard (DI) is needed.

Table 34 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 34 – Namespaces used in this document

NamespaceURI	Namespace Index	Example
http://opcfoundation.org/UA/	0	0:EngineeringUnits
http://opcfoundation.org/UA/DI/	2	2:DeviceRevision
http://opcfoundation.org/UA/Dictionary/IRDI/	3	3:0112/2///61987#xzx608

Annex A
(normative)

<Title> Namespace and mappings

A.1 Namespace and identifiers for <Title> Information Model

This appendix defines the numeric identifiers for all of the numeric *NodeIds* defined in this document. 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 <*type*> *ObjectType Node* which has the <*property*> *Property*. The **Name** for the <*property*> *InstanceDeclaration* within the <*type*> declaration is: *AutoIdDeviceType_DeviceLocation*.

A NamespaceURI follows the convention: http://opcfoundation.org/UA/<short name>/.
<short name> is described in 13.
.

Note that NamespaceURIs are NOT live URLs. Text in the specification should not suggest that they are.

The *NamespaceUri* for all *NodeIds* defined here is http://opcfoundation.org/UA/<short name>/

File Locations
The location of any version dependent files follow this convention:
http://opcfoundation.org/UA/schemas/<short name>/<version>/<file name>
The <short name> is the same as specified in the NamespaceURI;
The <version> is a number with the form #.# or #.##;
The location of the version independent files are the same but with the <version> omitted.
e.g. http://opcfoundation.org/UA/schemas/<short name>/<file name>

File Names
NodeIds: Opc.Ua.<short name>.NodeIds.csv or <short name>.NodeIds.csv

NodeSet: Opc.Ua.<short name>.NodeSet.xml or <short name>.NodeSet.xml;
Any other files should have a prefix that provides context when the file is downloaded in a browser.
All published files must be added to GitHub https://github.com/OPCFoundation/UA-Nodeset
This can be done by creating a mantis issue in the "NodeSets, XSDs and Generated Code" project:
https://opcfoundation-onlineapplications.org/mantis/main_page.php
The files should be attached to the mantis issue.
If the NodeSet was generated with the Opc.Ua.ModelCompiler the design file should be attached as well.

The CSV released with this version of the specification can be found here:
http://www.opcfoundation.org/UA/schemas/<short name>/1.0/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/<short name>/NodeIds.csv

A NodeIds.csv file is not mandated but recommended.
It contains a flat list of NodeIds with unique names and can be used instead of a full NodeSet if only such NodeId constants for a programming environment are needed.

A computer processible version of the complete Information Model defined in this document 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 (including any revisions, amendments or errata) can be found here:
http://www.opcfoundation.org/UA/schemas/<short name>/1.0/Opc.Ua.<short name>.NodeSet2.xml

NOTE The latest Information Model schema that is compatible with this version of the document can be found here:
http://www.opcfoundation.org/UA/schemas/short name>/Opc.Ua.<short name>.NodeSet2.xml