|  |  |  |
| --- | --- | --- |
| **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 UA Companion Specification |
| OPC nnnnn-m | |
| OPC UA for <Title>  Part <n>: <Part Name>  Release 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 <n> :<Part Name> | Date: | YYYY-MM-DD |
|  |  |  |  |
| Version: | Release 1.xy | Software: | MS-Word |
|  |  | Source: | OPC nnnnn-m - Title page for Joint Working Groups.docx |
|  |  |  |  |
| Author: | <author organization> | Status: | Release |
|  |  |  |  |

**CONTENTS**

Page

**Figures** iii

**Tables** iv

1 Scope 7

2 Normative references 7

3 Terms, definitions and conventions 7

3.1 Overview 7

3.2 OPC UA for <title> terms 8

3.3 Abbreviations and symbols 8

3.4 Conventions used in this document 8

3.4.1 Conventions for Node descriptions 8

3.4.2 NodeIds and BrowseNames 9

3.4.2.1 NodeIds 9

3.4.2.2 BrowseNames 10

3.4.3 Common Attributes 10

3.4.3.1 General 10

3.4.3.2 Objects 10

3.4.3.3 Variables 11

3.4.3.4 VariableTypes 11

3.4.3.5 Methods 11

4 General information to <title> and OPC UA 11

4.1 Introduction to <title> 11

4.2 Introduction to OPC Unified Architecture 11

4.2.1 What is OPC UA? 12

4.2.2 Basics of OPC UA 12

4.2.3 Information modelling in OPC UA 13

4.2.3.1 Concepts 13

4.2.3.2 Namespaces 16

4.2.3.3 Companion Specifications 16

5 Use cases 17

6 <title> Information Model overview 17

7 OPC UA ObjectTypes 17

7.1 <some>Type ObjectType Definition 17

7.1.1 Overview 17

8 OPC UA EventTypes 17

8.1.1 <some>EventType 17

9 OPC UA VariableTypes 18

9.1 <some>VariableType 18

10 OPC UA DataTypes 19

10.1 <someType> 19

11 Profiles and Namespaces 20

11.1 Namespace Metadata 20

11.2 Conformance Units and Profiles 20

11.3 Server Facets 20

11.4 Client Facets 21

11.5 Handling of OPC UA Namespaces 21

Annex A (normative): <Title> Namespace and mappings 23

A.1 Namespace and identifiers for <Title> Information Model 23

A.2 Profile URIs for <Title> Information Model 24

**Figures**

Figure 1 – The Scope of OPC UA within an Enterprise 12

Figure 2 – A Basic Object in an OPC UA Address Space 13

Figure 3 – The Relationship between Type Definitions and Instances 14

Figure 4 – Examples of References between Objects 15

Figure 5 – The OPC UA Information Model Notation 15

**Tables**

[Table 1 – Examples of DataTypes 9](#_Toc531777606)

[Table 2 – Type Definition Table 9](#_Toc531777607)

[Table 3 – Common Node Attributes 10](#_Toc531777608)

[Table 4 – Common Object Attributes 11](#_Toc531777609)

[Table 5 – Common Variable Attributes 11](#_Toc531777610)

[Table 6 – Common VariableType Attributes 11](#_Toc531777611)

[Table 7 – Common Method Attributes 11](#_Toc531777612)

[Table 8 – <some>Type Definition 17](#_Toc531777613)

[Table 9 – <some>EventType Definition 17](#_Toc531777614)

[Table 10 – <some>Type Definition 18](#_Toc531777615)

[Table 11 – SetPoints Structure 19](#_Toc531777616)

[Table 12 – <someType> Definition 19](#_Toc531777617)

[Table 13 – NamespaceMetadata Object for this Specification 20](#_Toc531777618)

[Table 14 – *Template Server Facet* Definition 20](#_Toc531777619)

[Table 15 – *Template Client Facet* Definition 21](#_Toc531777620)

[Table 16 – Namespaces used in a <title> Server 21](#_Toc531777621)

[Table 17 – Namespaces used in this specification 22](#_Toc531777622)

[Table 18 – Profile URIs 24](#_Toc531777623)

<OPC Foundation (if joint work)>, <other organization>

\_\_\_\_\_\_\_\_\_\_\_\_

[Public Domain Mark](https://creativecommons.org/publicdomain/mark/1.0/)  
This work (OPC UA Companion Specification Template, by [OPC Foundation](https://opcfoundation.org/)), identified by [OPC Foundation](https://opcfoundation.org/), is free of known copyright restrictions.

<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](https://opcfoundation.org/)”

>

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.>

COPYRIGHT RESTRICTIONS

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

PATENTS

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

WARRANTY AND LIABILITY DISCLAIMERS

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

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

RESTRICTED RIGHTS LEGEND

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

COMPLIANCE

The combination of <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.

# Scope

<Specify what this specification 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>

# Normative references

The following referenced documents are indispensable for the application of this specification. 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 10000-3: 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 10000-8: OPC Unified Architecture - Part 8: Data Access

<http://www.opcfoundation.org/UA/Part8/>

# Terms, definitions and conventions

## Overview

It is assumed that basic concepts of OPC UA information modelling and <other specifications> are understood in this specification. This specification 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, <other specifications> as well as the following apply.

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

## OPC UA for <title> terms

The following terms (1 and 2) are examples. They have the IEC format for term definitions.

term\_1

<a short description – max two lines>

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

term\_2

….

## Abbreviations and symbols

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

AC Alarm and Condition

DCS Distributed Control Systems

## Conventions used in this document

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

### 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** | **Data­Type** | **Value­Rank** | **Array­Dimensions** | **Description** |
| Int32 | Int32 | -1 | omitted or null | A scalar Int32. |
| Int32[] | Int32 | 1 | omitted or {0} | Single-dimensional array of Int32 with an unknown size. |
| Int32[][] | Int32 | 2 | omitted or {0,0} | Two-dimensional array of Int32 with unknown sizes for both dimensions. |
| Int32[3][] | Int32 | 2 | {3,0} | Two-dimensional array of Int32 with a size of 3 for the first dimension and an unknown size for the second dimension. |
| Int32[5][3] | Int32 | 2 | {5,3} | Two-dimensional array of Int32 with a size of 5 for the first dimension and a size of 3 for the second dimension. |
| Int32{Any} | Int32 | -2 | omitted or null | An Int32 where it is unknown if it is scalar or array with any number of dimensions. |
| Int32{ScalarOrOneDimension} | Int32 | -3 | omitted or null | An Int32 where it is either a single-dimensional array or a scalar. |

* 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.

*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** | **ModellingRule** |
| *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*. | Referenced *ModellingRule* of the referenced *Object*. |
| NOTE Notes referencing footnotes of the table content. | | | | | |

Components of *Nodes* can be complex that is containing components by themselves. The *TypeDefinition*, *NodeClass*, *DataType* and *ModellingRule* can be derived from the type definitions, and the symbolic name can be created as defined in 3.4.3.1. Therefore, those containing components are not explicitly specified; they are implicitly specified by the type definitions.

### NodeIds and BrowseNames

#### 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 specification 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 specification, the symbolic name is unique.

The NamespaceUri for all *NodeIds* defined in this specification 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 specification 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 specification, because they are not defined by this specification but generated by the *Server*.

#### BrowseNames

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

If the *BrowseName* is not defined by this specification, 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 specification. provides a list of namespaces and their indexes as used in this specification.

### Common Attributes

#### 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 3 shall be set as specified in the table.

Table 3 – Common Node Attributes

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

#### Objects

For all *Objects* specified in this specification, the *Attributes* named in Table 4 shall be set as specified in the Table 4. The definitions for the *Attributes* can be found in OPC 10000-3.

Table 4 – Common Object Attributes

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

#### Variables

For all *Variables* 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 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. |

#### VariableTypes

For all *VariableTypes* 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 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*. |

#### Methods

For all *Methods* 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 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. |

# General information to <title> and OPC UA

## Introduction to <title>

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

## Introduction to OPC Unified Architecture

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

### 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.

### Basics of OPC UA

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

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

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



Figure 1 – The Scope of OPC UA within an Enterprise

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

### Information modelling in OPC UA

#### Concepts

OPC UA provides a framework that can be used to represent complex information as *Objects* in an *AddressSpace* which can be accessed with standard services. These *Objects* consist of *Nodes* connected by *References*. Different classes of *Nodes* convey different semantics. For example, a *Variable Node* represents a value that can be read or written. The *Variable Node* has an associated *DataType* that can define the actual value, such as a string, float, structure etc. It can also describe the *Variable* value as a variant. A *Method Node* represents a function that can be called. Every *Node* has a number of *Attributes* including a unique identifier called a *NodeId* and non-localized name called as *BrowseName*. An *Object* representing a ‘Reservation’ is shown in Figure 2.



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

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

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

OPC UA also supports the concept of 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 expected 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)

#### 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. Namespaces in OPC UA have a globally unique string called a NamespaceUri and a locally unique integer called a NamespaceIndex. The NamespaceIndex is only unique within the context of a *Session* between an OPC UA *Client* and an OPC UA *Server*. The *Services* defined for OPC UA use the NamespaceIndex to specify the Namespace for qualified values.

There are two types of values in OPC UA that are qualified with Namespaces: NodeIds and QualifiedNames. NodeIds are globally unique identifiers for *Nodes*. This means the same *Node* with the same NodeId can appear in many *Servers*. This, in turn, means Clients can have built in knowledge of some *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.

#### 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.

# Use cases

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

# <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.

# OPC UA ObjectTypes

## <some>Type ObjectType Definition

### Overview

The <some>*Type* provides … and is formally defined in Table 8.

Table 8 – <some>Type Definition

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Attribute** | | **Value** | | | | |
| BrowseName | | <some>Type | | | | |
| IsAbstract | | False | | | | |
| **References** | **Node Class** | | **BrowseName** | **DataType** | **TypeDefinition** | **Modelling Rule** |
| Subtype of the <other>Type defined in …, i.e. inheriting the InstanceDeclarations of that Node. | | | | | | |
| HasProperty | Variable | | <PropertyName1> | String | PropertyType | Mandatory |
| HasProperty | Variable | | <PropertyName1> | Int32 | PropertyType | Mandatory |

*…*

# OPC UA EventTypes

### <some>EventType

This *EventType* is ….. Its representation in the *AddressSpace* is formally defined in Table 9.

Table 9 – <some>EventType Definition

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

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

# OPC UA VariableTypes

## <some>VariableType

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

It is formally defined in Table 10.

Table 10 – <some>Type Definition

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

# OPC UA DataTypes

## <someType>

This structure contains …. The structure is defined in Table 11.

Table 11 – SetPoints Structure

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| <someType> | structure |  |
| SP1 | Byte[] | Setpoint 1 |
| SP2 | Byte[] | Setpoint 2 |

Its representation in the *AddressSpace* is defined in Table 12.

Table 12 – <someType> Definition

|  |  |
| --- | --- |
| **Attributes** | **Value** |
| BrowseName | <someType> |

# Profiles and Namespaces

## Namespace Metadata

Namespace Metadata are required for any companion standard that specifies an information model (e.g. Objects and Object Types). The metadata provide standardized information about the elements of this namespace. This information is particularly important for aggregating Servers.

Table 13 defines the namespace metadata for this specification. 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 Part5 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 Part5.

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

Table 13 – NamespaceMetadata Object for this Specification

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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.y |
| HasProperty | NamespacePublicationDate | | DateTime | yyyy-mm-dd |
| HasProperty | IsNamespaceSubset | | Boolean | True or False |
| HasProperty | StaticNodeIdTypes | | IdType[] |  |
| HasProperty | StaticNumericNodeIdRange | | NumericRange[] |  |
| HasProperty | StaticStringNodeIdPattern | | String |  |

## Conformance Units and Profiles

This chapter defines the corresponding *Profiles* and *Conformance Units* for the OPC UA Information Model for <title>. *Profiles* are named groupings of *Conformance Units*. *Facets* are *Profiles* that will be combined with other *Profiles* to define the complete functionality of an OPC UA *Server* or *Client.*

## Server Facets

The following tables specify the *Facets* available for *Servers* that implement the <title> Information Model 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.

The following table is a template for a facet.

Table 14 defines afacet for the minimum functionality necessary ….

Table 14 – *Template Server Facet* Definition

| **Conformance Unit** | **Description** | **Optional/**  **Mandatory** |
| --- | --- | --- |
| CU 1 | Supports …. | M |
| CU 2 | Supports …. | M |
| CU 3 | Supports …. | O |
| **Profile** | | |
| ComplexType Server Facet (defined in OPC UA Part 7) | | M |
| BaseDevice\_Server\_Facet (defined in OPC UA Part 100) | | M |

## Client Facets

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

The following table is a template for a facet.

Table 15 defines afacet for the minimum functionality necessary for ….

Table 15 – *Template Client Facet* Definition

| **Conformance Unit** | **Description** | **Optional/**  **Mandatory** |
| --- | --- | --- |
| CU 1 | Uses …. | M |
| CU 2 | Uses …. | M |
| CU 3 | Uses …. | O |
| **Profile** | | |
| Method Client Facet (defined in OPC UA Part 7) | | M |
| BaseDevice\_Client\_Facet (defined in OPC UA Part 100) | | M |

## 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 specification shall not use the standard namespaces.

Table 16 provides a list of mandatory and optional namespaces used in an <title> OPC UA *Server*.

Table 16 – 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 |
| 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 UA Part 100. The namespace index is *Server* specific. | Mandatory |
| http://opcfoundation.org/UA/<title>/ | Namespace for *NodeIds* and *BrowseNames* defined in this specification. 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 specification 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 17 provides a list of namespaces and their index used for *BrowseNames* in this specification. The default namespace of this specification is not listed since all *BrowseNames* without prefix use this default namespace.

Table 17 – Namespaces used in this specification

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

1. (normative): <Title> Namespace and mappings
   1. Namespace and identifiers for <Title> 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 *<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>/](http://opcfoundation.org/UA/POWERLINK/) where the short name is all caps if an acronym or camel case if words.

Exception if the short name is a trademark. Use trademark casing.

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>/](http://opcfoundation.org/UA/%3cshort%20name%3e/)

**File Locations**

The location of any version dependent files follow this convention:

[http://opcfoundation.org/UA/schemas/<short name>/<version>/<file name>](http://opcfoundation.org/UA/schemas/%3cshort%20name%3e/%3cversion%3e/%3cfile%20name%3e)

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>](http://opcfoundation.org/UA/schemas/%3cshort%20name%3e/%3cfile%20name%3e)

**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](http://www.opcfoundation.org/UA/schemas/%3cshort%20name%3e/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](http://www.opcfoundation.org/UA/schemas/%3cshort%20name%3e/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 specification is also provided. It follows the XML Information Model schema syntax defined in Part 6.

The Information Model Schema released with this version of the specification can be found here:

[http://www.opcfoundation.org/UA/schemas/<short name>/1.0/Opc.Ua.<short name>.NodeSet2.xml](http://www.opcfoundation.org/UA/schemas/%3cshort%20name%3e/1.0/Opc.Ua.%3cshort%20name%3e.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/short name>/Opc.Ua.<short name>.NodeSet2.xml](http://www.opcfoundation.org/UA/schemas/short%20name%3e/Opc.Ua.%3cshort%20name%3e.NodeSet2.xml)

* 1. Profile URIs for <Title> Information Model

Table A.6 defines the Profile URIs for the <title> Information Model companion specification.

Table A.6 – Profile URIs

|  |  |
| --- | --- |
| **Profile** | **Profile URI** |
| First facet | http://opcfoundation.org/UA-Profile/External/<short name>/<first facet name> |
| Second facet | http://opcfoundation.org/UA-Profile/External/<short name>/<second facet name> |
| … |  |

\_\_\_\_\_\_\_\_\_\_\_