|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | | **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 <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: |  |
|  |  |  |  |
| Doc-Number | OPC nnnnn-m |  |  |
| Title: | :<Part Name> | Date: |  |
|  |  |  |  |
| Version: |  | Software: | MS-Word |
|  |  | Source: | OPC 20020(11020) - UA Companion Specification Template v1.01.17.docx |
|  |  |  |  |
| Author: |  | Status: | Draft |
|  |  |  |  |

**Template Revisions**

|  |  |  |
| --- | --- | --- |
| Version | Date | Description |
| 1.01.09 | Mar 09, 2020 | [M-4665](https://apps.opcfoundation.org/mantis/view.php?id=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](https://apps.opcfoundation.org/mantis/view.php?id=5314)). |
| 1.01.11 | July 09, 2020 | [M-5595](https://apps.opcfoundation.org/mantis/view.php?id=5595): Added better wording and recommendations for NamespaceMetadata as suggested. [M-5520](https://apps.opcfoundation.org/mantis/view.php?id=5520): Fixed quotation marks. [M-5781](https://apps.opcfoundation.org/mantis/view.php?id=5781): Title "Additional Variable Attributes" misleading. [M-5673](https://apps.opcfoundation.org/mantis/view.php?id=5673): Added HasInterface Reference to figure. [M-5468](https://apps.opcfoundation.org/mantis/view.php?id=5468): Use of Namespace Index also in Method signatures. [M-5789](https://apps.opcfoundation.org/mantis/view.php?id=5789): Clarify use of NamespaceIndex for Profiles or ConformanceUnits in other specifications. |
| 1.01.12 | Nov 16, 2020 | [M-5904](https://apps.opcfoundation.org/mantis/view.php?id=5904): BrowseName convention for placeholders. [M-6184](https://apps.opcfoundation.org/mantis/view.php?id=6184): The "additional attributes" table shall not be restricted to Value and Description. [M-6183](https://apps.opcfoundation.org/mantis/view.php?id=6183): Fix heading names for columns with a browse path. [M-6112](https://apps.opcfoundation.org/mantis/view.php?id=6112): Add an example state machine. [M-6185](https://apps.opcfoundation.org/mantis/view.php?id=6185): Add new table types to conventions. |
| 1.01.13 | Feb 17, 2021 | [M-6302](https://apps.opcfoundation.org/mantis/view.php?id=6302): Updated format for NamespaceMetadata table to make it validatable. [M-6317](https://apps.opcfoundation.org/mantis/view.php?id=6317): Added guideline for uncaptioned embedded figures. |
| 1.01.14 | Nov 15, 2021 | [M-6264](https://apps.opcfoundation.org/mantis/view.php?id=6264): GeneratesEvent example added. [M-7147](https://apps.opcfoundation.org/mantis/view.php?id=7147): Added abbreviated terms used in template. [M-6265](https://apps.opcfoundation.org/mantis/view.php?id=6265): No HasSubtype References. [M-7265](https://apps.opcfoundation.org/mantis/view.php?id=7265): Added max size for ConformanceUnit description. [M-6612](https://apps.opcfoundation.org/mantis/view.php?id=6612): Added definition of capability ID (Annex A.2). [M-6150](https://apps.opcfoundation.org/mantis/view.php?id=6150): Extended tables with Conformance Units row. |
| 1.01.15 | Oct 01, 2022 | [M-7528](https://apps.opcfoundation.org/mantis/view.php?id=7528): Namespace index also for the “Subtype of” rows. [M-8025](https://apps.opcfoundation.org/mantis/view.php?id=8025): Added missing BrowsePath tables for subcomponents. [M-6776](https://apps.opcfoundation.org/mantis/view.php?id=6776): Changed the examples for Structure and Union to start with an uppercase character. [M-7341](https://mantis.opcfoundation.org/view.php?id=7341): Added extended definition for structures. |
| 1.01.16 | Mar 14, 2023 | [M-8638](https://apps.opcfoundation.org/mantis/view.php?id=8638): Updated URL format for schema files (Annex A). Updated URL format for normative references to documents (Clause 2). |
| 1.01.17 |  | [M-9196](https://apps.opcfoundation.org/mantis/view.php?id=9196): Conformance Units and Profiles should be in profile database. [M-8765](https://apps.opcfoundation.org/mantis/view.php?id=8765): Added URL syntax for individual files. [M-9170](https://apps.opcfoundation.org/mantis/view.php?id=9170): Casing of UANodeSet wrong. [M-9182](https://apps.opcfoundation.org/mantis/view.php?id=9182): Added reference to IEC specifications. |

**CONTENTS**

[1 Scope 1](#_Toc114584674)

[2 Normative references 1](#_Toc114584675)

[3 Terms, abbreviated terms and conventions 3](#_Toc114584676)

[3.1 Overview 3](#_Toc114584677)

[3.2 OPC UA for <title> terms 3](#_Toc114584678)

[3.3 Abbreviated terms 3](#_Toc114584679)

[3.4 Conventions used in this document 4](#_Toc114584680)

[3.4.1 Conventions for Node descriptions 4](#_Toc114584681)

[3.4.2 NodeIds and BrowseNames 7](#_Toc114584682)

[3.4.3 Common Attributes 8](#_Toc114584683)

[3.4.4 Structures 9](#_Toc114584684)

[4 EDITING Guidelines 11](#_Toc114584685)

[5 General information to <title> and OPC UA 12](#_Toc114584686)

[5.1 Introduction to <title> 12](#_Toc114584687)

[5.2 Introduction to OPC Unified Architecture 12](#_Toc114584688)

[5.2.1 What is OPC UA? 12](#_Toc114584689)

[5.2.2 Basics of OPC UA 12](#_Toc114584690)

[5.2.3 Information modelling in OPC UA 13](#_Toc114584691)

[6 Use cases 17](#_Toc114584692)

[7 <title> Information Model overview 18](#_Toc114584693)

[8 OPC UA ObjectTypes 18](#_Toc114584694)

[8.1 <some>Type definition 18](#_Toc114584695)

[8.1.1 Overview 18](#_Toc114584696)

[8.1.2 ObjectType definition 18](#_Toc114584697)

[8.1.3 <some>Method 21](#_Toc114584698)

[8.2 ExampleStateMachineType 22](#_Toc114584699)

[9 OPC UA EventTypes 24](#_Toc114584700)

[9.1 <some>EventType 24](#_Toc114584701)

[10 OPC UA VariableTypes 24](#_Toc114584702)

[10.1 <some>VariableType 24](#_Toc114584703)

[11 OPC UA DataTypes 24](#_Toc114584704)

[11.1 <someStructure> 24](#_Toc114584705)

[11.2 <someUnion> 25](#_Toc114584706)

[11.3 <someEnumeration> 25](#_Toc114584707)

[11.4 <someOptionSet> 26](#_Toc114584708)

[12 OPC UA ReferenceTypes 26](#_Toc114584709)

[12.1 <someReferenceType> 26](#_Toc114584710)

[13 Instances 27](#_Toc114584711)

[13.1 <someInstance> 27](#_Toc114584712)

[14 Profiles and Conformance Units 28](#_Toc114584713)

[14.1 Conformance Units 28](#_Toc114584714)

[14.2 Profiles 28](#_Toc114584715)

[14.2.1 Profile list 28](#_Toc114584716)

[14.2.2 Server Facets 29](#_Toc114584717)

[14.2.3 Client Facets 30](#_Toc114584718)

[15 Namespaces 30](#_Toc114584719)

[15.1 Namespace Metadata 30](#_Toc114584720)

[15.2 Handling of OPC UA Namespaces 31](#_Toc114584721)

[Annex A (normative) <Title> Namespace and mappings 33](#_Toc114584722)

[A.1 Namespace and identifiers for <Title> Information Model 33](#_Toc114584723)

[A.2 Capability Identifier 34](#_Toc114584724)

**Figures**

[Figure 1 – The scope of OPC UA within an enterprise 11](#_Toc85097736)

[Figure 2 – A basic Object in an OPC UA Address Space 12](#_Toc85097737)

[Figure 3 – The relationship between Type Definitions and Instances 13](#_Toc85097738)

[Figure 4 – Examples of References between Objects 14](#_Toc85097739)

[Figure 5 – The OPC UA Information Model notation 14](#_Toc85097740)

**Tables**

[Table 1 – Examples of DataTypes iv](#_Toc85097741)

[Table 2 – Type Definition Table v](#_Toc85097742)

[Table 3 – Examples of Other Characteristics v](#_Toc85097743)

[Table 4 – <some>Type Additional References vi](#_Toc85097744)

[Table 5 – <some>Type Additional Subcomponents vi](#_Toc85097745)

[Table 6 – <some>Type Attribute values for child Nodes vi](#_Toc85097746)

[Table 7 – Common Node Attributes vii](#_Toc85097747)

[Table 8 – Common Object Attributes viii](#_Toc85097748)

[Table 9 – Common Variable Attributes viii](#_Toc85097749)

[Table 10 – Common VariableType Attributes viii](#_Toc85097750)

[Table 11 – Common Method Attributes viii](#_Toc85097751)

[Table 12 – <Table With Embedded Graphics> ix](#_Toc85097752)

[Table 13 – <some>Type definition 16](#_Toc85097753)

[Table 14 – <some>Type Additional References 17](#_Toc85097754)

[Table 15 – <some>Type Additional Subcomponents 18](#_Toc85097755)

[Table 16 – <some>Type Attribute values for child Nodes 19](#_Toc85097756)

[Table 17 – <some>Method Method Arguments 20](#_Toc85097757)

[Table 18 – <some>Method Method AddressSpace definition 20](#_Toc85097758)

[Table 19 – ExampleStateMachineType definition 21](#_Toc85097759)

[Table 20 – ExampleStateMachineType Additional References 21](#_Toc85097760)

[Table 21 – ExampleStateMachineType Attribute values for child Nodes 21](#_Toc85097761)

[Table 22 – <some>EventType definition 22](#_Toc85097762)

[Table 23 – <some>Type definition 22](#_Toc85097763)

[Table 24 – <someStructure> Structure 22](#_Toc85097764)

[Table 25 – <someStructure> Definition 23](#_Toc85097765)

[Table 26 – <someUnion> Union 23](#_Toc85097766)

[Table 27 – <someUnion> definition 23](#_Toc85097767)

[Table 28 – <someEnumeration> Items 23](#_Toc85097768)

[Table 29 – <someEnumeration> Definition 24](#_Toc85097769)

[Table 30 – <someOptionSet> Values 24](#_Toc85097770)

[Table 31 – <someOptionSet> definition 24](#_Toc85097771)

[Table 32 – <someReferenceType> definition 25](#_Toc85097772)

[Table 33 – <someInstance> definition 25](#_Toc85097773)

[Table 34 – Conformance Units for <Title> 26](#_Toc85097774)

[Table 35 – Profile URIs for <Title> 26](#_Toc85097775)

[Table 36 - <short name> <Prf1name> Server Profile 27](#_Toc85097776)

[Table 37 - <short name> <Prf2name> Server Facet 27](#_Toc85097777)

[Table 38 - <short name> < Prf3name> Client Facet 28](#_Toc85097778)

[Table 39 – NamespaceMetadata Object for this Document 29](#_Toc85097779)

[Table 40 – Namespaces used in a <title> Server 29](#_Toc85097780)

[Table 41 – Namespaces used in this document 29](#_Toc85097781)

<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.

**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.>

Scope shall specify the type of change. The three options are **Errata** (corrected something that was wrong), **Clarification** (make something clearer that was not wrong), and **Feature** (added new functionality).

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

| **Mantis ID** | **Scope** | **Summary** | **Resolution** |
| --- | --- | --- | --- |
| [3165](http://opcfoundation-onlineapplications.org/mantis/view.php?id=3165) | Errata | Wrong variable data types in Program example. | Fixed datatypes and replaced Table A.14 by description of the variables. |
| [3521](http://opcfoundation-onlineapplications.org/mantis/view.php?id=3521) | Clarification | 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

# 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, authorisation, 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 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.

NOTE The OPC UA core specifications are regularly published as IEC 62541.

<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/documents/10000-1/>

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

<http://www.opcfoundation.org/documents/10000-2/>

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

<http://www.opcfoundation.org/documents/10000-3/>

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

<http://www.opcfoundation.org/documents/10000-4/>

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

<http://www.opcfoundation.org/documents/10000-5/>

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

<http://www.opcfoundation.org/documents/10000-6/>

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

<http://www.opcfoundation.org/documents/10000-7/>

OPC 10000-8, *OPC Unified Architecture - Part 8: Data Access*

<http://www.opcfoundation.org/documents/10000-8/>

OPC 10000-9, *OPC Unified Architecture - Part 9: Alarms and Conditions*

<http://www.opcfoundation.org/documents/10000-9/>

OPC 10000-10, *OPC Unified Architecture - Part 10: Programs*

<http://www.opcfoundation.org/documents/10000-10/>

OPC 10000-11, *OPC Unified Architecture - Part 11: Historical Access*

<http://www.opcfoundation.org/documents/10000-11/>

OPC 10000-12, *OPC Unified Architecture - Part 12: Discovery and Global Services*

<http://www.opcfoundation.org/documents/10000-12/>

OPC 10000-13, *OPC Unified Architecture - Part 13: Aggregates*

<http://www.opcfoundation.org/documents/10000-13/>

OPC 10000-14, *OPC Unified Architecture - Part 14: PubSub*

<http://www.opcfoundation.org/documents/10000-14/>

OPC 10000-15, *OPC Unified Architecture - Part 15: Safety*

<http://www.opcfoundation.org/documents/10000-15/>

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

<http://www.opcfoundation.org/documents/10000-16/>

OPC 10000-17, *OPC Unified Architecture - Part 17: Alias Names*

<http://www.opcfoundation.org/documents/10000-17/>

OPC 10000-18, *OPC Unified Architecture - Part 18: Role-Based Security*

<http://www.opcfoundation.org/documents/10000-18/>

OPC 10000-19, *OPC Unified Architecture - Part 19: Dictionary References*

<http://www.opcfoundation.org/documents/10000-19/>

OPC 10000-20, *OPC Unified Architecture - Part 20: File Transfer*

<http://www.opcfoundation.org/documents/10000-20/>

OPC 10000-22, *OPC Unified Architecture - Part 22: Base Network Model*

<http://www.opcfoundation.org/documents/10000-22/>

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

<http://www.opcfoundation.org/documents/10000-100/>

OPC 10000-110, *OPC Unified Architecture - Part 110: Asset Management Basics*

<http://www.opcfoundation.org/documents/10000-110/>

OPC 10000-200, *OPC Unified Architecture - Part 200: Industrial Automation*

<http://www.opcfoundation.org/documents/10000-200/>

Examples for references to other companion specifications

OPC 40001-1, *OPC UA for Machinery - Part 1: Basic Building Blocks*

<http://www.opcfoundation.org/documents/40001-1/>

OPC 10031-4, *OPC UA for ISA-95 – Part 4: Job Control*

<http://www.opcfoundation.org/documents/10031-4/>

# Terms, abbreviated terms and conventions

## 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.

## OPC UA for <title> terms

Term descriptions are expected to be written in such a form that the term definition can replace the term in its context.

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]

term 2

definition 2

## Abbreviated terms

The following abbreviations are examples and those used in this template. The list shall only contain abbreviations used in the document.

AC Alarm and Condition

DCS Distributed Control Systems

ERP Enterprise Resource Planning

HMI Human Machine Interface

HTTP Hypertext Transfer Protocol

IP Internet Protocol

MES Manufacturing Execution System

PLC Programable Logical Controller

PMS Production Management System

TCP Transmission Control Protocol

UML Unified Modelling Language

URI Uniform Resource Identifier

XML Extensible Markup Language

## Conventions used in this document

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

### Conventions for Node descriptions

#### Node definitions

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

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

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

* If the *TargetNode* is a component of the *Node* being defined in the table the *Attributes* of the composed *Node* are defined in the same row of the table.
* The *DataType* is only specified for *Variables*; “[<number>]” indicates a single-dimensional array, for multi-dimensional arrays the expression is repeated for each dimension (e.g. [2][3] for a two-dimensional array). For all arrays the *ArrayDimensions* is set as identified by <number> values. If no <number> is set, the corresponding dimension is set to 0, indicating an unknown size. If no number is provided at all the *ArrayDimensions* can be omitted. If no brackets are provided, it identifies a scalar *DataType* and the *ValueRank* is set to the corresponding value (see OPC 10000-3). In addition, *ArrayDimensions* is set to null or is omitted. If it can be Any or *ScalarOrOneDimension*, the value is put into “{<value>}”, so either “{Any}” or “{*ScalarOrOneDimension*}” and the *ValueRank* is set to the corresponding value (see OPC 10000-3) and the *ArrayDimensions* is set to null or is omitted. Examples are given in Table 1.

Table 1 – Examples of DataTypes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Notation** | **Data­Type** | **Value­Rank** | **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.

Each *Type* *Node* or well-known *Instance Node* defined shall have one or more *ConformanceUnits* defined in 14.1 that require the *Node* to be in the *AddressSpace*.

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

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

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

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

For additional details see OPC 10000-5.

Table 2 – Type Definition Table

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

Components of *Nodes* can be complex that is containing components by themselves. The *TypeDefinition*, *NodeClass* and *DataType* can be derived from the type definitions, and the symbolic name can be created as defined in 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.

#### Additional References

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

Table 4 – <some>Type additional References

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

*References* can be to any other *Node*.

#### Additional sub-components

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

Table 5 – <some>Type additional subcomponents

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

#### Additional Attribute values

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

Table 6 – <some>Type Attribute values for child Nodes

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

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

### 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 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*.

#### 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 15.2.

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

If a *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 41 provides a list of namespaces and their indexes as used in this document.

### 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 document or if it is server-specific.

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

Table 7 – Common Node Attributes

|  |  |
| --- | --- |
| **Attribute** | **Value** |
| DisplayName | The *DisplayName* is a *LocalizedText*. Each *Server* shall provide the *DisplayName* identical to the *BrowseName* of the *Node* for the *LocaleId* “en” unless specified differently in the specification. 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 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. |

#### Objects

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

Table 8 – Common Object Attributes

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

#### Variables

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

Table 9 – Common Variable Attributes

|  |  |
| --- | --- |
| **Attribute** | **Value** |
| MinimumSamplingInterval | Optionally, a server-specific minimum sampling interval is provided. |
| AccessLevel | The access level for *Variables* used for type definitions is server-specific, for all other *Variables* defined in this 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. |

#### VariableTypes

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

Table 10 – Common VariableType Attributes

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

#### Methods

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

Table 11 – Common Method Attributes

|  |  |
| --- | --- |
| **Attributes** | **Value** |
| Executable | All *Methods* defined in this 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. |

### Structures

OPC 10000-3 differentiates between different kinds of *Structures*. The following conventions explain, how these *Structures* shall be defined.

The first kind are *Structures* without optional fields where none of the fields allows subtype (except fields with abstract *DataTypes*). Its definition is in Table 12.

Table 12 – Structures without optional fields where none of the fields allow subtypes

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

The second kind are *Structures* with optional fields where none of the fields allows subtypes (except fields with abstract *DataTypes*). Its definition is in Table 13.

Structures with fields that are optional have an “Optional” column. Fields that are optional have True set, otherwise False.

Table 13 – Structures with optional fields

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Description | Optional |
| <someStructure> | structure | Subtype of <someParentStructure> defined in … |  |
| SP1 | 0:Byte[] | Setpoint 1 | False |
| Optional Field\_1 | 0:String | Some Text | True |

The third kind are *Structures* without optional fields where one or more of the fields allow subtypes. Its definition is in Table 14.

Structures with fields that allow subtypes have an “Allow Subtypes” column. Fields that allow subtypes have True set, otherwise False. Fields with abstract *DataTypes* can always be subtyped.

Table 14 – Structures where one or more of the fields allow subtypes

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Description | Allow Subtypes |
| <someStructure> | structure | Subtype of <someParentStructure> defined in … |  |
| SP1 | 0:Byte[] | Setpoint 1 | False |
| Allow Subtypes | 0:ByteString | Some Bytestring | True |

Fields with abstract *DataTypes* shall have True in the “Allow Subtypes” column.

It is not allowed to add both columns to combine optional fields and fields that allow subtypes in one structure.

# EDITING Guidelines

This section provides overall guidelines for editors. **Please delete before publication.**

**Guideline 1: FIGURES**

Figures shall be embedded document objects (Powerpoint, Excel or Visio).

Figures shall be followed by a caption (Figure <n> - <Description>).

**\*\* Do not use inline Word drawing objects! \*\***

**\*\* This includes blocks of code which are captioned as figures! \*\***

The style for Figures is “**FIGURE**”

The style for Figure Captions is “**FIGURE-title**”

If it is necessary to add an embedded object to the document that is not a figure that appears in the TOC then the style shall be “**FIGURE-uncaptioned**”. Table 12 provides an example of uncaptioned embedded objects.

Table 12 – <Table With Embedded Graphics>

|  |  |  |
| --- | --- | --- |
| NodeClass | Graphical Representation | Comment |
| Object |  | Rectangle including text representing the string-part of the *DisplayName* of the *Object*. The font shall not be set to italic. |
| Variable |  | Rectangle with rounded corners including text representing the string-part of the *DisplayName* of the *Variable*. The font shall not be set in italic. |

**Guideline 2: Heading capitalization**

This guideline is applied for all OPC UA parts following the IEC guidelines:

The first letter is capitalized.

For headers that consist of multiple words, all other words are all lower case with the exception of proper nouns, like terms or type names.

This applies to section, table, and figure headers.

**Guideline 3: Do not use HasSubtype References in Type definition tables**

HasSubtype References conflict with the ConformanceUnit References and therefore should be avoided. All HasSubtype References have been removed from the existing types defined in Part 5 (Mantis [6650](https://mantis.opcfoundation.org/view.php?id=6650)).

# 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 visualisation 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 subtype 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.

The 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 functionalities (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. 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.

#### 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 definition

### Overview

An overview description of <some>Type. This can be supplemented with a type model figure.

The <some>*Type* provides … and is formally defined in Table 13.

### ObjectType definition

Table 13 – <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.  e.g., Subtype of the 0:BaseObjectType defined in OPC 10000-5 … | | | | | |
| 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 | Variable | SimpleArray | 0:Int32[] | 0:BaseDataVariableType | O |
| 0:HasComponent | Variable | ComplexArray | 0:EnumValueType[] | 0:BaseDataVariableType | O |
| 0:HasComponent | Object | <some>Alarm |  | 0:AlarmConditionType | O |
| 0:HasComponent | Method | <some>Method | See 8.1.2 | | M |
| 0:HasDictionaryEntry | Object | 3:0112/2///61987#xzx607 |  |  |  |
| 0:GeneratesEvent | ObjectType | 0:DeviceFailureEventType |  |  |  |
| **Conformance Units** | | | | | |
| <ConformanceUnit\_1> | | | | | |
| <ConformanceUnit\_2> | | | | | |
| <ConformanceUnit\_3> | | | | | |

|  |
| --- |
| SPECIFYING CONFORMANCE UNITS  The *ConformanceUnits* listed in the table are the primary *ConformanceUnits* used to test the capabilities provided by a *Node*. *ConformanceUnits* which indirectly require the *Node* are not listed.  For example, a *ConformanceUnit* testing a subtype of a *ObjectType* does not need to be listed since a supertype is automatically included if a subtype is included.  If there are no suitable *ConformanceUnits* then the *Node* may not be needed or there are missing *ConformanceUnits.*  The OPC Foundation tooling used to validate the specification during the release process automatically adds the *ConformanceUnits* listed to the *Category* element for the *Node* definition in the UANodeSet. Therefore, editors do not have to add these *Category* elements manually into the UANodeSet.  See also corresponding description in 3.4.1.1. |

Here should be text for the description of the table rows.

<some>Property1 is used for …

The <some>Alarm becomes active, if …

The components of the <some>*Type* have additional references which are defined in Table 14.

Table 14 – <some>Type additional References

|  |  |  |  |
| --- | --- | --- | --- |
| **SourceBrowsePath** | **Reference Type** | **Is Forward** | **TargetBrowsePath** |
| <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 | |

|  |
| --- |
| SPECIFYING ADDITIONAL REFERENCES  <See 3.4.1.2 for the definition of this table.>  Table 13 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 14. This format allows you to add *References* from those *InstanceDeclarations* to any other *Node*.  In the following figure shows of the *AddressSpace* as defined by Table 13 and Table 14.    SPECIFYING DICTIONARY REFERENCES  When making use of dictionary references (see OPC 10000-19) the following rules apply:   * *DictionaryEntries* from a *TypeDefinition* shall be referenced directly (see Table 13) * *DictionaryEntries* from *InstanceDeclarations* shall be referenced by using an ‘Additional References’ table for the *TypeDefinition* (see Table 14). * When using IRDIs or URIs   + use the well-defined namespace (see OPC 10000-19) and include this in Table 41. 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 subcomponents which are defined in Table 15.

Table 15 – <some>Type additional subcomponents

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **BrowsePath** | **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  <See 3.4.1.3 for the definition of this table.>  Table 13 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 subcomponents, you can use a table format as shown in Table 15. This format allows you to add instances to those *InstanceDeclarations*.  In the following figure shows of the *AddressSpace* as defined by Table 13 and Table 15. |

The child *Nodes* of the <some>*Type* have additional *Attribute* values defined in Table 16.

|  |
| --- |
| <See 3.4.1.4 for the definition of this table.>  It allows to specify *Attributes* like the Value or Description that cannot be specified in the *ObjectType* table.  Fields may be empty which means this *Attribute* is not defined. A column where all fields are empty should be omitted. |

Table 16 – <some>Type Attribute values for child Nodes

|  |  |  |
| --- | --- | --- |
| **BrowsePath** | **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 |
| SimpleArray | [0,1,2] |  |
| ComplexArray | [  {“Value”: 0, "DisplayName": " ON ",  "Description": “some description”},  {"Value": 1, "DisplayName": "OFF",  "Description": "some other description"}  ] |  |

### <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 17 and Table 18 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 17 – <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 18 – <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 |

## ExampleStateMachineType

This section gives an example of how a *StateMachine* is defined using the table formats used in the temple. This allows the Validator to check the correctness of the *StateMachine* in the Spec with the NodeSet-File

Here needs to be text describing the *StateMachine*. In our case, we use a very simple *StateMachine* with two states. A figure is always helpful.



Figure 6 Example State Machine

The *ExampleStateMachineType* provides … and is formally defined in Table 19.

Table 19 – ExampleStateMachineType definition

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Attribute** | **Value** | | | | | |
| BrowseName | ExampleStateMachineType | | | | | |
| IsAbstract | False | | | | | |
| **References** | **Node Class** | **BrowseName** | **DataType** | **TypeDefinition** | **Other** | |
| Subtype of the 0:FiniteStateMachineType defined in …, i.e. inheriting the InstanceDeclarations of that Node. | | | | | | |
| 0:HasComponent | Object | Idle |  | 0:InitialStateType |  | |
| 0:HasComponent | Object | Working |  | 0:StateType |  | |
| 0:HasComponent | Object | FromIdleToWorking |  | 0:TransitionType |  | |
| 0:HasComponent | Object | FromWorkingToIdle |  | 0:TransitionType |  | |
| 0:HasComponent | Method | Start |  |  | M | |
| 0:HasComponent | Method | Stop |  |  | M | |
| **Conformance Units** | | | | | |
| <ConformanceUnit\_4> | | | | | |
| <ConformanceUnit\_5> | | | | | | |

The components of the *ExampleStateMachineType* have additional references which are defined in Table 20.

Table 20 – ExampleStateMachineType additional References

|  |  |  |  |
| --- | --- | --- | --- |
| **SourceBrowsePath** | **Reference Type** | **Is Forward** | **TargetBrowsePath** |
| FromIdleToWorking | 0:FromState | True | Idle |
|  | 0:ToState | True | Working |
|  | 0:HasCause | True | Start |
| FromWorkingToIdle | 0:FromState | True | Working |
|  | 0:ToState | True | Idle |
|  | 0:HasCause | True | Stop |
|  | 0:HasEffect | True | 0:BaseEventType |

Note: Method signature would need to be defined as well.

The component *Variables* of the *ExampleStateMachineType* have additional *Attributes* defined in Table 21.

Table 21 – ExampleStateMachineType Attribute values for child Nodes

|  |  |
| --- | --- |
| **BrowsePath** | **Value Attribute** |
| |  | | --- | | Idle | | 0:StateNumber | | 1 |
| |  | | --- | | Working | | 0:StateNumber | | 2 |
| |  | | --- | | FromIdleToWorking | | 0:TransitionNumber | | 1 |
| |  | | --- | | FromWorkingToIdle | | 0:TransitionNumber | | 2 |

# OPC UA EventTypes

## <some>EventType

This *EventType* is ….. Its representation in the *AddressSpace* is formally defined in Table 22.

Table 22 – <some>EventType definition

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Attribute** | | **Value** | | | | |
| BrowseName | | <some>EventType | | | | |
| IsAbstract | | True | | | | |
| **References** | **NodeClass** | | **BrowseName** | **DataType** | **TypeDefinition** | **Other** |
| Subtype of the 0:*BaseEventType* defined in …, which means it inherits the InstanceDeclarations of that Node. | | | | | | |
| 0:HasProperty | Variable | | <some>Eventfield | 0:String | 0:PropertyType | 0:Mandatory |
| **Conformance Units** | | | | | | |
| <ConformanceUnit\_6> | | | | | | |
| <ConformanceUnit\_7> | | | | | | |

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 23.

Table 23 – <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 0:BaseDataVariableType defined in … | | | | | | |
| 0:HasComponent | Variable | | <var1> | 0:UtcTime | 0:BaseDataVariableType | 0:Mandatory |
| 0:HasComponent | Variable | | <var2> | 0:UtcTime | 0:BaseDataVariableType | 0:Mandatory |
| **Conformance Units** | | | | | | |
| <ConformanceUnit\_8> | | | | | | |
| <ConformanceUnit\_9> | | | | | | |

# OPC UA DataTypes

## <someStructure>

See 3.4.4 for the different kinds of Structures.

This structure contains …. The structure is defined in Table 24.

Table 24 – <someStructure> Structure (with field that allows subtypes)

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Description | Allow Subtypes |
| <someStructure> | structure | Subtype of <someParentStructure> defined in … |  |
| SP1 | 0:Byte[] | Setpoint 1 | False |
| SP2 | 0:Byte[] | Setpoint 2 | False |
| Allow Subtypes | 0:ByteString | Some Bytestring | True |

Its representation in the *AddressSpace* is defined in Table 25.

Table 25 – <someStructure> definition

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Attribute** | | **Value** | | | | |
| BrowseName | | <someStructure> | | | | |
| IsAbstract | | False | | | | |
| **References** | **NodeClass** | | **BrowseName** | **DataType** | **TypeDefinition** | **Other** |
| Subtype of the <someParentStructure> defined in … | | | | | | |
| **Conformance Units** | | | | | | |
| <ConformanceUnit\_10> | | | | | | |
| <ConformanceUnit\_11> | | | | | | |

## <someUnion>

This union contains …. The union is defined in Table 26.

Table 26 – <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 27.

Table 27 – <someUnion> definition

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Attributes** | | **Value** | | | | |
| BrowseName | | <someUnion> | | | | |
| IsAbstract | | False | | | | |
| **References** | **NodeClass** | | **BrowseName** | **DataType** | **TypeDefinition** | **Other** |
| Subtype of 0:Union defined in OPC 10000-5. | | | | | | |
| **Conformance Units** | | | | | | |
| <ConformanceUnit\_12> | | | | | | |
| <ConformanceUnit\_13> | | | | | | |

## <someEnumeration>

This enumeration …. The enumeration is defined in Table 28.

Table 28 – <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 29.

Table 29 – <someEnumeration> definition

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Attribute** | | **Value** | | | | |
| BrowseName | | <someEnumeration> | | | | |
| IsAbstract | | False | | | | |
| **References** | **NodeClass** | | **BrowseName** | **DataType** | **TypeDefinition** | **Other** |
| Subtype of the 0:Enumeration type defined in OPC 10000-5 | | | | | | |
| 0:HasProperty | Variable | | 0:EnumStrings | 0:LocalizedText [] | 0:PropertyType |  |
| **Conformance Units** | | | | | | |
| <ConformanceUnit\_14> | | | | | | |
| <ConformanceUnit\_15> | | | | | | |

## <someOptionSet>

This *DataType* defines flags for … *<*someOptionSet*>* is formally defined in Table 30.

Table 30 – <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 31.

Table 31 – <someOptionSet> definition

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Attribute** | | **Value** | | | | |
| BrowseName | | <someOptionSet> | | | | |
| IsAbstract | | False | | | | |
| **References** | **NodeClass** | | **BrowseName** | **DataType** | **TypeDefinition** | **Other** |
| Subtype of the 0:OptionSet DataType defined in OPC 10000-5 | | | | | | |
| 0:HasProperty | Variable | | 0:OptionSetValues | 0:LocalizedText [] | 0:PropertyType |  |
| **Conformance Units** | | | | | | |
| <ConformanceUnit\_16> | | | | | | |
| <ConformanceUnit\_17> | | | | | | |

# OPC UA ReferenceTypes

## <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 32.

Table 32 – <someReferenceType> definition

|  |  |  |  |
| --- | --- | --- | --- |
| **Attributes** | **Value** | | |
| BrowseName | <someReferenceType> | | |
| InverseName | <someinverseName> | | |
| Symmetric | <True/False> | | |
| IsAbstract | <True/False> | | |
| **References** | **NodeClass** | **BrowseName** | **Comment** |
| Subtype <someParentReferenceType> | | | |
| **Conformance Units** | | | |
| <ConformanceUnit\_18> | | | |
| <ConformanceUnit\_19> | | | |

# Instances

## <someInstance>

The *<*someInstance*>* is formally defined in Table 33.

Table 33 – <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> | |
| **Conformance Units** | | | | |
| <ConformanceUnit\_20> | | | | |
| <ConformanceUnit\_21> | | | | |

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

# Profiles and Conformance Units

*Profiles* and *ConformanceUnits* break functionality into testable groups. All companion specification shall include at least one *Profile*/*Facet*.

*Profiles* and *ConformanceUnits* are configured in the Profile Database.

See OPC 20023 for guidelines to do this.

**<short name>**

A <short name> is required for each companion specification to assure uniqueness of string identifiers. It precedes the names of Profiles and ConformanceUnits 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.

Meaning and significance of *Profiles* and *ConformanceUnits* are described in OPC 10000-7.

The *Profiles* and *ConformanceUnits* for this specification are maintained in an online database and accessible via [https://profiles.opcfoundation.org/pg=<name of profile group>](https://profiles.opcfoundation.org/?%3cname%20of%20profile%20group%3e/).

The ProfileGroup name is specified in OPC 20023 and can be viewed via the GUI of the profile database.

Examples:

OPC 10000-200 (IA): <https://profiles.opcfoundation.org/?pg=IA-GM%201.0x>

OPC 40000-nnn: <https://profiles.opcfoundation.org/?pg=Machinery%201.x>

OPC 10000-100 (DI): <https://profiles.opcfoundation.org/?pg=DI%201.04>

A snapshot of the profile group (with profiles and conformance units) can be included here as well. For this, the profile database provides an application to export all information of a profile group as a word document. The contents of this word document can then be pasted in this chapter.

An introductory paragraph shall describe whether this is a snapshot (database contains always the latest information) or it is the formal definition (database is just a better GUI).

# Namespaces

## 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 39 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 39 – NamespaceMetadata Object for this document

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Value** | | |
| BrowseName | [http://opcfoundation.org/UA/<short](http://opcfoundation.org/UA/%3cshort) name>/ | | |
| **Property** | | **DataType** | **Value** |
| NamespaceUri | | String | [http://opcfoundation.org/UA/<short](http://opcfoundation.org/UA/%3cshort) name> |
| NamespaceVersion | | String | X.YY.ZZ |
| NamespacePublicationDate | | DateTime | YYYY-MM-DD |
| IsNamespaceSubset | | Boolean | False |
| StaticNodeIdTypes | | IdType[] | 0 |
| StaticNumericNodeIdRange | | NumericRange [] |  |
| StaticStringNodeIdPattern | | String |  |

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

## 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 40 provides a list of namespaces typically used in a <title> OPC UA *Server*.

Table 40 – Namespaces used in a <title> Server

| **NamespaceURI** | **Description** |
| --- | --- |
| http://opcfoundation.org/UA/ | Namespace for *NodeIds* and *BrowseNames* defined in the OPC UA specification. This namespace shall have namespace index 0. |
| Local Server URI | Namespace for nodes defined in the local server. This namespace shall have namespace index 1. |
| http://opcfoundation.org/UA/DI/ | Namespace for *NodeIds* and *BrowseNames* defined in OPC 10000-100. The namespace index is *Server* specific. |
| http://opcfoundation.org/UA/<title>/ | Namespace for *NodeIds* and *BrowseNames* defined in this document. The namespace index is *Server* specific. |
| Vendor specific types | A *Server* may provide vendor-specific types like types derived from *ObjectTypes* defined in this document in a vendor-specific namespace. |
| 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. |

Table 41 provides a list of namespaces and their indices 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 41 – 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 |

1. (normative)   
     
   <Title> Namespace and mappings
   1. NodeSet and supplementary files for <Title> Information Model

An *Information Model* is formally defined in an XML file called a *NodeSet* This file conforms to the standard syntax defined in the Annex “Information Model XML Schema” OPC 10000-6. It can be read and processed by a computer program.

An *Information Model* is identified by a URI – the so-called *NamespaceUri*.

A *NamespaceUri* follows one of these conventions:

http://opcfoundation.org/UA/<short name>/

tag:opcfoundation.org,yyyy-MM:UA:<short name>

Where <short name> is described in 14 and yyyy-MM is the date when the *NamespaceUri* was first published. *NamespaceUris* are not network accessible URLs and the text should not suggest they are. The tag URI syntax allows authors to choose a URI that cannot be used as URL by mistake. Note that the date in the tag syntax is not the same as the *PublicationDate* for the *NodeSet*. It is set once when the URI is created and never changed.

The Online Reference provides a summary page for every *NamespaceUri* of released *Information Models* which has the form:

[https://reference.opcfoundation.org/nodesets?u=<NamespaceUri>](https://reference.opcfoundation.org/nodesets?u=%3cNamespaceUri%3e)

The <Title> *Information Model* is identified by the following URI:

http://opcfoundation.org/UA/<short name>/

Documentation for the NamespaceUri can be found [here](https://reference.opcfoundation.org/nodesets?u=%3cNamespaceUri%3e).

“here” is a hyperlink to the summary page of the Online Reference.

In the hyperlink, the <NamespaceUri> has to be replaced with the concrete URI for this specification.

The *NodeSet* associated with this version of specification can be found here:

https://reference.opcfoundation.org/nodesets/?u=<NamespaceUri>&v=<Version>&i=1

The *NodeSet* associated with the latest version of the specification can be found here:

https://reference.opcfoundation.org/nodesets/?u=<NamespaceUri>&i=1

<NamespaceUri> is the *NamespaceUri* for the *Information Model*.

The <Version> is the string in the *NamespaceVersion* from the *Namespace* *Metadata* (see 15.1). This value is also the value of the *Version* attribute in the *NodeSet*.

Supplementary files for the <Title> *Information Model* can be found here:

https://reference.opcfoundation.org/nodesets/?u=<NamespaceUri>&v=<Version>&i=2

The files associated with the latest version of the specification can be found here:

https://reference.opcfoundation.org/nodesets/?u=<NamespaceUri>&i=2

Supplementary files should be provided when appropriate (i.e., IRDI NodeSets or examples)

**File Names**

**NodeIds**: Opc.Ua.<short name>.NodeIds.csv

**NodeSet**: Opc.Ua.<short name>.NodeSet2.xml;

Any other files should have a prefix that provides context when the file is downloaded in a browser.

**When DataTypes are defined in the NodeSet**:

Opc.ua.<short name>.Types.xsd –

The XML schema for the DataTypes defined by the specification;

Opc.ua.<short name>.Types.bsd –

The OPC Binary schema for the DataTypes defined by the specification;

In case there is the need to reference individual files the following syntax should be added to the template:

https://reference.opcfoundation.org/files/<file>?u=<namespaceUri>&v=<version>  
https://reference.opcfoundation.org/files/<file>?u=<namespaceUri>

for example:

https://reference.opcfoundation.org/files/StatusCode.csv?u=http://opcfoundation.org/UA/&v=1.05.02  
https://reference.opcfoundation.org/files/StatusCode.csv?u=http://opcfoundation.org/UA/  
https://reference.opcfoundation.org/files/Opc.Ua.Types.xsd?u=http://opcfoundation.org/UA/&v=1.05.02  
https://reference.opcfoundation.org/files/Opc.Ua.Types.xsd?u=http://opcfoundation.org/UA/

* 1. Capability Identifier

*ServerCapabilityIdentifiers* are defined in OPC 10000-12. They can be used for features, like certain information models, which are likely to be useful during the discovery process. The identifiers shall be short because of length restrictions for fields used in the mDNS specification.

The identifier shall be up to 6 characters. It is recommended to use the <short name> introduced in the guideline, at the beginning of clause 14 if that meets the requirement of up to 6 characters.

Note, that such identifiers are not required. If not needed, this Annex section shall be deleted.

The capability identifier for this document shall be:

<short name>

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