|  |  |  |
| --- | --- | --- |
|  |  | OPC UA Companion Specification |
| OPC 40563-4 | |
| OPC UA for Mining - Transport Dumping  Part 4: Conveying  Working Draft 1.0  2025-10-01  **OPC 40563-4 (Working Draft 1.0) is identical with VDMA 40563-4:2025-10** | |

**Contents**

Page

[Forewords 8](#_Toc194572919)

[1 Scope 9](#_Toc194572920)

[2 Normative references 9](#_Toc194572921)

[3 Terms, definitions and conventions 11](#_Toc194572922)

[3.1 Overview 11](#_Toc194572923)

[3.2 OPC UA for <title> terms 11](#_Toc194572924)

[3.3 Abbreviated terms 11](#_Toc194572925)

[3.4 Conventions used in this document 11](#_Toc194572926)

[3.4.1 Conventions for Node descriptions 11](#_Toc194572927)

[3.4.2 NodeIds and BrowseNames 14](#_Toc194572928)

[3.4.3 Common Attributes 15](#_Toc194572929)

[3.4.4 Structures 16](#_Toc194572930)

[4 General information to <title> and OPC UA 19](#_Toc194572931)

[4.1 Introduction to <title> 19](#_Toc194572932)

[4.2 Introduction to OPC Unified Architecture 19](#_Toc194572933)

[4.2.1 What is OPC UA? 19](#_Toc194572934)

[4.2.2 Basics of OPC UA 19](#_Toc194572935)

[4.2.3 Information modelling in OPC UA 20](#_Toc194572936)

[5 Use cases 24](#_Toc194572937)

[6 <title> Information Model overview 24](#_Toc194572938)

[7 OPC UA ObjectTypes 24](#_Toc194572939)

[7.1 BeltConveyorType ObjectType Definition 24](#_Toc194572940)

[7.1.1 Overview 24](#_Toc194572941)

[7.2 DriveControllerType ObjectType Definition 26](#_Toc194572942)

[7.2.1 Overview 26](#_Toc194572943)

[7.2.2 ConveyorStart 26](#_Toc194572944)

[7.2.3 ConveyorStop 27](#_Toc194572945)

[7.2.4 CloseBreak 27](#_Toc194572946)

[7.2.5 OpenBreak 28](#_Toc194572947)

[7.3 ConveyorDriveType ObjectType Definition 28](#_Toc194572948)

[7.3.1 Overview 28](#_Toc194572949)

[7.4 ConveyorBrakeType ObjectType Definition 31](#_Toc194572950)

[7.4.1 Overview 31](#_Toc194572951)

[7.5 BeltType ObjectType Definition 32](#_Toc194572952)

[7.5.1 Overview 32](#_Toc194572953)

[7.6 BeltSectionType ObjectType Definition 33](#_Toc194572954)

[7.6.1 Overview 33](#_Toc194572955)

[7.7 BeltSpliceType ObjectType Definition 34](#_Toc194572956)

[7.7.1 Overview 34](#_Toc194572957)

[7.8 PulleyType ObjectType Definition 35](#_Toc194572958)

[7.8.1 Overview 35](#_Toc194572959)

[7.9 WeatherStationType ObjectType Definition 36](#_Toc194572960)

[7.9.1 Overview 36](#_Toc194572961)

[7.10 ScraperType ObjectType Definition 37](#_Toc194572962)

[7.10.1 Overview 37](#_Toc194572963)

[7.11 ConveyorRollType ObjectType Definition 38](#_Toc194572964)

[7.11.1 Overview 38](#_Toc194572965)

[8 Profiles and ConformanceUnits 39](#_Toc194572966)

[9 Namespaces 39](#_Toc194572967)

[9.1 Namespace Metadata 39](#_Toc194572968)

[9.2 Handling of OPC UA Namespaces 39](#_Toc194572969)

[Annex A (normative) <Title> Namespace and mappings 41](#_Toc194572970)

**Figures**

[Figure 1 – The Scope of OPC UA within an Enterprise 20](#_Toc194403808)

[Figure 2 – A Basic Object in an OPC UA Address Space 21](#_Toc194403809)

[Figure 3 – The Relationship between Type Definitions and Instances 22](#_Toc194403810)

[Figure 4 – Examples of References between Objects 23](#_Toc194403811)

[Figure 5 – The OPC UA Information Model Notation 23](#_Toc194403812)

**Tables**

[Table 1 – Examples of DataTypes 12](#_Toc194403813)

[Table 2 – Type Definition Table 13](#_Toc194403814)

[Table 3 – Examples of Other Characteristics 13](#_Toc194403815)

[Table 4 – <some> Additional References 13](#_Toc194403816)

[Table 5 – <some>Type Additional Subcomponents 14](#_Toc194403817)

[Table 6 – <some>Type Attribute values for child nodes 14](#_Toc194403818)

[Table 7 – Common Node Attributes 15](#_Toc194403819)

[Table 8 – Common Object Attributes 15](#_Toc194403820)

[Table 9 – Common Variable Attributes 16](#_Toc194403821)

[Table 10 – Common VariableType Attributes 16](#_Toc194403822)

[Table 11 – Common Method Attributes 16](#_Toc194403823)

[Table 12 – Structures without optional fields where none of the fields allow subtypes 16](#_Toc194403824)

[Table 13 – Structures with optional fields 17](#_Toc194403825)

[Table 14 – Structures where one or more of the fields allow subtypes 17](#_Toc194403826)

[Table 41 – NamespaceMetadata Object for this Document 39](#_Toc194403827)

[Table 42 – Namespaces used in a <title> Server 40](#_Toc194403828)

[Table 43 – Namespaces used in this document 40](#_Toc194403829)

**OPC Foundation / VDMA**

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

**AGREEMENT OF USE**

COPYRIGHT RESTRICTIONS

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

PATENTS

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

WARRANTY AND LIABILITY DISCLAIMERS

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

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

RESTRICTED RIGHTS LEGEND

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

COMPLIANCE

The combination of VDMA and OPC Foundation shall at all times be the sole entities that may authorize developers, suppliers and sellers of hardware and software to use certification marks, trademarks or other special designations to indicate compliance with these materials as specified within this document. Products developed using this specification may claim compliance or conformance with this specification if and only if the software satisfactorily meets the certification requirements set by VDMA or the OPC Foundation. Products that do not meet these requirements may claim only that the product was based on this specification and must not claim compliance or conformance with this specification.

TRADEMARKS

Most computer and software brand names have trademarks or registered trademarks. The individual trademarks have not been listed here.

GENERAL PROVISIONS

Should any provision of this Agreement be held to be void, invalid, unenforceable or illegal by a court, the validity and enforceability of the other provisions shall not be affected thereby.

This Agreement shall be governed by and construed under the laws of Germany.

This Agreement embodies the entire understanding between the parties with respect to, and supersedes any prior understanding or agreement (oral or written) relating to, this specification.

ISSUE REPORTING

If an error or problem is found in this specification, the UaNodeSet, or any associated supplementary files, it should be reported as an issue.

The reporting process can be found here: <https://opcfoundation.org/resources/issue-tracking/>

The Link to the issue tracking project for this document is here:

[https://mantis.opcfoundation.org/set\_project.php?project\_id=<nnn>&make\_default=no](https://mantis.opcfoundation.org/set_project.php?project_id=%3cnnn%3e&make_default=no)

<nnn> is the project\_id in Mantis which is created for any document when requested by the working group. Example: <https://mantis.opcfoundation.org/set_project.php?project_id=142&make_default=no> is the Link for OPC 40001-\* (Machinery).

If you have no Mantis Project or do not know the project\_id, please send a request to [TechnicalDirector@opcfoundation.org](mailto:TechnicalDirector@opcfoundation.org).

Forewords

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

|  |  |
| --- | --- |
| **Version** | **Changes** |
| 1.0.0 | Initial Release |
|  |  |

OPC UA is a machine to machine communication technology to transmit characteristics of products (e.g. manufacturer name, device type or components) and process data (e.g. temperatures, pressures or feed rates). To enable vendor unspecific interoperability the description of product characteristics and process data has to be standardized utilizing technical specifications, the OPC UA companion specifications.

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

OPC Foundation

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

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

* Platform independence: from an embedded microcontroller to cloud-based infrastructure
* Secure: encryption, authentication, authorization and auditing
* Extensible: ability to add new features including transports without affecting existing applications
* Comprehensive information modelling capabilities: for defining any model from simple to complex

VDMA Mining

The VDMA represents over 3,200 mainly small and medium size member companies in the engineering industry, making it one of the largest and most important industrial associations in Europe. With an export quota amounting to 96 per cent, mining technology is one of the most export-oriented branches of the German engineering industry. VDMA Mining represents well-known, mainly medium-sized companies from the sectors open cast mining/materials handling, underground mining, mining processing technology and consulting, research and development. 145 companies merged in VDMA Mining representing more than 90 per cent of the entire trade volume.

# 

MAIN TITLE IN CAPITAL LETTERS –

Part X: Second part of the title in normal letters

# Scope

This document XXXXX specifies / establishes / ...

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

# Normative references

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

There are no normative references in this document.

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, 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, … 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.

4.2.1

term 1

<a short description – max two lines>

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

EXAMPLE 1 First example for term 1.

EXAMPLE 2 Second example for term 1.

[SOURCE: where definition 1 was found]

4.2.2

term 2

definition 2

## Abbreviated terms

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

AC Alarm and Condition

DCS Distributed Control Systems

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{OneOrMoreDimensions} | 0:Int32 | 0 | omitted or null | The Int32 value is an array with one or more dimensions. |
| 0:Int32[] | 0:Int32 | 1 | omitted or {0} | Single-dimensional array of Int32 with an unknown size. |
| 0:Int32[][] | 0:Int32 | 2 | omitted or {0,0} | Two-dimensional array of Int32 with unknown sizes for both dimensions. |
| 0:Int32[3][] | 0:Int32 | 2 | {3,0} | Two-dimensional array of Int32 with a size of 3 for the first dimension and an unknown size for the second dimension. |
| 0:Int32[5][3] | 0:Int32 | 2 | {5,3} | Two-dimensional array of Int32 with a size of 5 for the first dimension and a size of 3 for the second dimension. |
| 0:Int32{Any} | 0:Int32 | -2 | omitted or null | An Int32 where it is unknown if it is scalar or array with any number of dimensions. |
| 0:Int32{ScalarOrOneDimension} | 0:Int32 | -3 | omitted or null | An Int32 where it is either a single-dimensional array or a scalar. |

* The TypeDefinition is specified for *Objects* and *Variables*.
* The TypeDefinition column specifies a symbolic name for a *NodeId*, i.e. the specified *Node* points with a *HasTypeDefinition* *Reference* to the corresponding *Node*.
* The *ModellingRule* of the referenced component is provided by specifying the symbolic name of the rule in the *ModellingRule* column. In the *AddressSpace*, the *Node* shall use a *HasModellingRule* *Reference* to point to the corresponding *ModellingRule* *Object*.

If the *NodeId* of a *DataType* is provided, the symbolic name of the *Node* representing the *DataType* shall be used.

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

*Nodes* of all other *NodeClasses* cannot be defined in the same table; therefore only the used *ReferenceType*, their *NodeClass* and their *BrowseName* are specified. A reference to another part of this document points to their definition. Table 2 illustrates the table. If no components are provided, the DataType, TypeDefinition and ModellingRule columns may be omitted and only a Comment column is introduced to point to the *Node* definition.

Each *Type* *Node* or well-known *Instance Node* defined shall have one or more *ConformanceUnits* defined in **Fehler! Verweisquelle konnte nicht gefunden werden.** 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 “--“ will be used. | | | | |
|  |  | | | | |
| **References** | **NodeClass** | **BrowseName** | **DataType** | **TypeDefinition** | **Other** |
| *ReferenceType* name | *NodeClass* of the target *Node*. | *BrowseName* of the target *Node*. | *DataType* of the referenced *Node*, only applicable for *Variables*. | *TypeDefinition* of the referenced *Node*, only applicable for *Variables* and *Objects*. | Additional characteristics of the *TargetNode* such as the *ModellingRule* or *AccessLevel*. |
| NOTE Notes referencing footnotes of the table content. | | | | | |
| **Conformance Units** | | | | | |
| Name of *ConformanceUnit*, one row per *ConformanceUnit* | | | | | |

Components of *Nodes* can be complex that is containing components by themselves. The *TypeDefinition*, *NodeClass* and *DataType* can be derived from the type definitions, and the symbolic name can be created as defined in 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> 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** | **Reference** | **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 9.2.

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 the *BrowseName* is not defined by this document, a namespace index prefix is added to the *BrowseName* (e.g., prefix '0' leading to ‘0:EngineeringUnits’ or prefix '2' leading to ‘2:DeviceRevision’). This is typically necessary if a *Property* of another specification is overwritten or used in the OPC UA types defined in this document. Table 55 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 specification or if it is server-specific.

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

Table 7 – Common Node Attributes

|  |  |
| --- | --- |
| **Attribute** | **Value** |
| DisplayName | The *DisplayName* is a *LocalizedText*. Each server shall provide the *DisplayName* identical to the *BrowseName* of the *Node* for the LocaleId “en”. Whether the server provides translated names for other LocaleIds is server-specific. |
| Description | Optionally a server-specific description is provided. |
| NodeClass | Shall reflect the *NodeClass* of the *Node.* |
| NodeId | The *NodeId* is described by *BrowseNames* as defined in 3.4.2.1. |
| WriteMask | Optionally the *WriteMask* *Attribute* can be provided. If the *WriteMask* *Attribute* is provided, it shall set all non-server-specific *Attributes* to not writable. For example, the *Description* *Attribute* may be set to writable since a *Server* may provide a server-specific description for the *Node*. The *NodeId* shall not be writable, because it is defined for each *Node* in this specification. |
| UserWriteMask | Optionally the *UserWriteMask* *Attribute* can be provided. The same rules as for the *WriteMask* *Attribute* apply. |
| RolePermissions | Optionally server-specific role permissions can be provided. |
| UserRolePermissions | Optionally the role permissions of the current Session can be provided. The value is server-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 8 shall be set as specified in the table. The definitions for the *Attributes* can be found in OPC 10000-3.

Table 8 – Common Object Attributes

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

#### Variables

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

Table 9 – Common Variable Attributes

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

#### VariableTypes

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

Table 10 – Common VariableType Attributes

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

#### Methods

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

Table 11 – Common Method Attributes

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

### 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 |
| SP2 | 0:Byte[] | Setpoint 2 | 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.

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

#### 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 one or two 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

## BeltConveyorType ObjectType Definition

### Overview

The *BeltConveyorType* describes the entry point into the information model of a belt conveyor. It is defined in Table 15.

Table – BeltConveyorType Definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute** | **Value** | | | | |
| BrowseName | BeltConveyorType | | | | |
| IsAbstract | False | | | | |
| **References** | **Node Class** | **BrowseName** | **DataType** | **TypeDefinition** | **Other** |
| Subtype of the *5:MiningEquipmentType* defined in OPC 40560 | | | | | |
| 0:HasAddIn | Object | MiningEquipmentIdentification |  | 5:MiningEquipmentIdentificationType | M |
| 0:HasComponent | Object | Components |  | 4:MachineComponentsType | O |
| 0:HasComponent | Object | Monitoring |  | 4:MonitoringType | M |
| 0:HasComponent | Object | MachineryBuildingBlocks |  | 0:FolderType | M |
| **Conformance Units** | | | | | |
|  | | | | | |
|  | | | | | |

*MiningEquipmentIdentification* represents numerous identification features of a mining equipment. It is a subtype of the *4:MachineIdentificationType*.

*Components* represents a collection of all identifiable components contained below this *Object*.

*Monitoring* represents a collection of *ObjectTypes* and *VariableTypes*, representing the current state of the process, that are not assigned to a component but to this *Object*.

*MachineryBuidlingBlocks* is representing a folder that directly references all those building blocks of the OPC UA for Machinery (OPC 40001-1, OPC 40001-3) which are implemented as an add-in.

The components of the *BeltConveyorType* have additional references which are defined in Table 16.

Table – BeltConveyorType Additional References

|  |  |  |  |
| --- | --- | --- | --- |
| **SourceBrowsePath** | **Reference Type** | **Is Forward** | **TargetBrowsePath** |
| MachineryBuildingBlocks | 0:HasAddIn | True | MiningEquipmentIdentification |
| MachineryBuildingBlocks | 0:HasAddIn | True | Components |
| MachineryBuildingBlocks | 0:HasAddIn | True | Monitoring |

The components of the *BeltConveyorType* have additional subcomponents which are defined in Table 17.

Table – BeltConveyorType Additional Subcomponents

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Source Path** | **Reference** | **NodeClass** | **BrowseName** | **DataType** | **TypeDefinition** | **Others** |
| Components | 0:HasComponent | Object | DriveController |  | DriveControllerType | O |
| Components | 0:HasComponent | Object | Belt |  | BeltType | O |
| Components | 0:HasComponent | Object | <Pulley> |  | PulleyType | OP |
| Components | 0:HasComponent | Object | <BeltScale> |  |  | OP |
| Components | 0:HasComponent | Object | <WeatherStation> |  | WeatherStationType | OP |
| Components | 0:HasComponent | Object | <Scraper> |  | ScraperType | OP |
| Components | 0:HasComponent | Object | <ConveyorRoll> |  | ConveyorRollType | OP |
| |  | | --- | | Monitoring | | Status | | 0:HasComponent | Object | ConveyorStateMachine |  |  | M |
| |  | | --- | | Monitoring | | Status | | 0:HasComponent |  | TypeOfRunning |  |  | O |

*DriveController* is

*Belt* is

*Pulley* is

*BeltScale* is

*WeatherStation* is

*Scraper* is

*ConveyorRollType* is

*ConveyorStateMachine* is

*TypeOfRunning* is

## DriveControllerType ObjectType Definition

### Overview

The *DriveControllerType* provides all the data required for a drive controller of a conveyor belt system that is needed to fulfil the defined use cases. It is formally defined in Table 18.

Table – DriveControllerType Definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute** | **Value** | | | | |
| BrowseName | DriveControllerType | | | | |
| IsAbstract | False | | | | |
| **References** | **Node Class** | **BrowseName** | **DataType** | **TypeDefinition** | **Other** |
| Subtype of the *5:MiningEquipmentType* defined in OPC 40560 | | | | | |
| 0:HasAddIn | Object | MiningEquipmentIdentification |  | 5:MiningEquipmentIdentificationType |  |
| 0:HasComponent | Object | Components |  | 4:MachineComponentsType |  |
| 0:HasComponent | Object | Operating |  | 0:FolderType |  |
| 0:HasComponent | Object | MachineryBuildingBlocks |  | 0:FolderType |  |
| **Conformance Units** | | | | | |
|  | | | | | |
|  | | | | | |

*MiningEquipmentIdentification* represents numerous identification features of a mining equipment. It is a subtype of the *4:MachineIdentificationType*.

*Components* represents a collection of all identifiable components contained below this *Object*.

*Operating* is representing a collection of *Methods* that can betriggered by this *Object*.

*Monitoring* represents a collection of *ObjectTypes* and *VariableTypes*, representing the current state of the process, that are not assigned to a component but to this *Object*.

*MachineryBuidlingBlocks* is representing a folder that directly references all those building blocks of the OPC UA for Machinery (OPC 40001-1, OPC 40001-3) which are implemented as an add-in.

The components of the *DriveControllerType* have additional references which are defined in Table 19.

Table – DriveControllerType Additional References

|  |  |  |  |
| --- | --- | --- | --- |
| **SourceBrowsePath** | **Reference Type** | **Is Forward** | **TargetBrowsePath** |
| MachineryBuildingBlocks | 0:HasAddIn | True | MiningEquipmentIdentification |
| MachineryBuildingBlocks | 0:HasAddIn | True | Components |

The components of the *DriveControllerType* have additional subcomponents which are defined in Table 20.

Table – DriveControllerType Additional Subcomponents

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Source Path** | **Reference** | **NodeClass** | **BrowseName** | **DataType** | **TypeDefinition** | **Others** |
| Operating | 0:HasComponent | Method | ConveyorStart |  |  |  |
| Operating | 0:HasComponent | Method | ConveyorStop |  |  |  |
| Operating | 0:HasComponent | Method | CloseBreak |  |  |  |
| Operating | 0:HasComponent | Method | OpenBreak |  |  |  |

### ConveyorStart

The *Method ConveyorStart* .... . The signature of this *Method* is specified below. Table 21 and Table 22 specify the *Arguments* and *AdressSpace* representation, respectively.

**Signature**

**ConveyorStart** (

[in] DataType InArg1,

[in] DataType InArg2,

[out] DataType OutArg1,

[out] DataType OutArg2)

Table 21 – ConveyorStart Method Arguments

|  |  |
| --- | --- |
| **Argument** | **Description** |
| InArg1 |  |
| InArg2 |  |
| OutArg1 |  |
| OutArg2 |  |

Table – ConveyorStart Method AddressSpace Definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute** | **Value** | | | | |
| BrowseName | ConveyorStart | | | | |
| **References** | **Node Class** | **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 |

### ConveyorStop

The *Method ConveyorStop* .... . The signature of this *Method* is specified below. Table 23 and Table 24 specify the *Arguments* and *AdressSpace* representation, respectively.

**Signature**

**ConveyorStop** (

[in] DataType InArg1,

[in] DataType InArg2,

[out] DataType OutArg1,

[out] DataType OutArg2)

Table 23 – ConveyorStop Method Arguments

|  |  |
| --- | --- |
| **Argument** | **Description** |
| InArg1 |  |
| InArg2 |  |
| OutArg1 |  |
| OutArg2 |  |

Table – ConveyorStop Method AddressSpace Definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute** | **Value** | | | | |
| BrowseName | ConveyorStop | | | | |
| **References** | **Node Class** | **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 |

### CloseBreak

The *Method CloseBreak* .... . The signature of this *Method* is specified below. Table 25 and Table 26 specify the *Arguments* and *AdressSpace* representation, respectively.

**Signature**

**CloseBreak** (

[in] DataType InArg1,

[in] DataType InArg2,

[out] DataType OutArg1,

[out] DataType OutArg2)

Table 25 – CloseBreak Method Arguments

|  |  |
| --- | --- |
| **Argument** | **Description** |
| InArg1 |  |
| InArg2 |  |
| OutArg1 |  |
| OutArg2 |  |

Table – CloseBreak Method AddressSpace Definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute** | **Value** | | | | |
| BrowseName | CloseBreak | | | | |
| **References** | **Node Class** | **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 |

### OpenBreak

The *Method OpenBreak* .... . The signature of this *Method* is specified below. Table 27 and Table 28 specify the *Arguments* and *AdressSpace* representation, respectively.

**Signature**

**OpenBreak** (

[in] DataType InArg1,

[in] DataType InArg2,

[out] DataType OutArg1,

[out] DataType OutArg2)

Table 27 – OpenBreak Method Arguments

|  |  |
| --- | --- |
| **Argument** | **Description** |
| InArg1 |  |
| InArg2 |  |
| OutArg1 |  |
| OutArg2 |  |

Table – OpenBreak Method AddressSpace Definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute** | **Value** | | | | |
| BrowseName | OpenBreak | | | | |
| **References** | **Node Class** | **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 |

## ConveyorDriveType ObjectType Definition

### Overview

The *ConveyorDriveType* provides ... and is formally defined in Table 21.

Table – ConveyorDriveType Definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute** | **Value** | | | | |
| BrowseName | ConveyorDriveType | | | | |
| IsAbstract | False | | | | |
| **References** | **Node Class** | **BrowseName** | **DataType** | **TypeDefinition** | **Other** |
| Subtype of the *5:MiningEquipmentType* defined in OPC 40560 | | | | | |
| 0:HasAddIn | Object | MiningEquipmentIdentification |  | 5:MiningEquipmentIdentificationType |  |
| 0:HasComponent | Object | Monitoring |  | 4:MonitoringType |  |
| 0:HasComponent | Object | MachineryBuildingBlocks |  | 0:FolderType |  |
| **Conformance Units** | | | | | |
|  | | | | | |
|  | | | | | |

*MiningEquipmentIdentification* represents numerous identification features of a mining equipment. It is a subtype of the *4:MachineIdentificationType*.

*Monitoring* represents a collection of *ObjectTypes* and *VariableTypes*, representing the current state of the process, that are not assigned to a component but to this *Object*.

*MachineryBuidlingBlocks* is representing a folder that directly references all those building blocks of the OPC UA for Machinery (OPC 40001-1, OPC 40001-3) which are implemented as an add-in.

The components of the *ConveyorDriveType* have additional references which are defined in Table 22.

Table – ConveyorDriveType Additional References

|  |  |  |  |
| --- | --- | --- | --- |
| **SourceBrowsePath** | **Reference Type** | **Is Forward** | **TargetBrowsePath** |
| MachineryBuildingBlocks | 0:HasAddIn | True | MiningEquipmentIdentification |
| MachineryBuildingBlocks | 0:HasAddIn | True | Monitoring |
| MachineryBuildingBlocks | 0:HasAddIn | True | |  | | --- | | Monitoring | | Status | | MachineryItemState | |

The components of the *ConveyorDriveType* have additional subcomponents which are defined in Table 23.

Table – ConveyorDriveType Additional Subcomponents

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Source Path** | **Reference** | **NodeClass** | **BrowseName** | **DataType** | **TypeDefinition** | **Others** |
| |  | | --- | | Monitoring | | Status | | 0:HasComponent | Object | MachineryItemState |  | 4:MachineryItemState\_StateMachineType |  |
| |  | | --- | | Monitoring | | Status | | 0:HasProperty | Variable | BeltRunningDirection |  | 0:Enumeration |  |
| |  | | --- | | Monitoring | | Status | | 0:HasProperty | Variable | SavedTorqueReached | 0:Boolean | 0:PropertyType |  |
| |  | | --- | | Monitoring | | Status | | 0:HasProperty | Variable | TargetSpeedReached | 0:Boolean | 0:PropertyType |  |
| |  | | --- | | Monitoring | | Status | | 0:HasProperty | Variable | ConveyorStopped | 0:Boolean | 0:PropertyType |  |
| |  | | --- | | Monitoring | | Status | | 0:HasProperty | Variable | AcknowledgeStart | 0:Boolean | 0:PropertyType |  |
| |  | | --- | | Monitoring | | Status | | 0:HasProperty | Variable | ConveyorStarted | 0:Boolean | 0:PropertyType |  |
| |  | | --- | | Monitoring | | Process | | 0:HasComponent | Variable | Torque | 0:Double | 0:AnalogUnitType |  |
| |  | | --- | | Monitoring | | Process | | 0:HasComponent | Variable | BeltSpeed | 0:Double | 0:AnalogUnitType |  |
| |  | | --- | | Monitoring | | Process | | 0:HasComponent | Variable | MotorSpeed | 0:Double | 0:AnalogUnitType |  |
| |  | | --- | | Monitoring | | Process | | 0:HasComponent | Variable | MotorPower | 0:Double | 0:AnalogUnitType |  |
| |  | | --- | | Monitoring | | Process | | 0:HasComponent | Variable | MotorCurrent | 0:Double | 0:AnalogUnitType |  |
| |  | | --- | | Monitoring | | Process | | 0:HasComponent | Variable | SavedTorque | 0:Double | 0:AnalogUnitType |  |

*MachineryItemState* is used as defined in OPC 40001-1. It represents a StateMachine that shows the current machine state.

*BeltRunningDirection* is

*SavedTorqueReached* is

*TargetSpeedReached* is

*ConveyorStopped* is

*AcknowledgeStart* is

*ConveyorStarted* is

*Torque* is

*BeltSpeed* is

*MotorSpeed* is

*MotorPower* is

*MotorCurrent* is

*SavedTorque* is

## ConveyorBrakeType ObjectType Definition

### Overview

The *ConveyorBrakeType* provides ... and is formally defined in Table 24.

Table – ConveyorBrakeType Definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute** | **Value** | | | | |
| BrowseName | ConveyorBrakeType | | | | |
| IsAbstract | False | | | | |
| **References** | **Node Class** | **BrowseName** | **DataType** | **TypeDefinition** | **Other** |
| Subtype of the *5:MiningEquipmentType* defined in OPC 40560 | | | | | |
| 0:HasAddIn | Object | MiningEquipmentIdentification |  | 5:MiningEquipmentIdentificationType |  |
| 0:HasComponent | Object | Monitoring |  | 4:MonitoringType |  |
| 0:HasComponent | Object | MachineryBuildingBlocks |  | 0:FolderType |  |
| **Conformance Units** | | | | | |
|  | | | | | |
|  | | | | | |

*MiningEquipmentIdentification* represents numerous identification features of a mining equipment. It is a subtype of the *4:MachineIdentificationType*.

*Monitoring* represents a collection of *ObjectTypes* and *VariableTypes*, representing the current state of the process, that are not assigned to a component but to this *Object*.

*MachineryBuidlingBlocks* is representing a folder that directly references all those building blocks of the OPC UA for Machinery (OPC 40001-1, OPC 40001-3) which are implemented as an add-in.

The components of the *ConveyorBrakeType* have additional references which are defined in Table 25.

Table – ConveyorBrakeType Additional References

|  |  |  |  |
| --- | --- | --- | --- |
| **SourceBrowsePath** | **Reference Type** | **Is Forward** | **TargetBrowsePath** |
| MachineryBuildingBlocks | 0:HasAddIn | True | MiningEquipmentIdentification |
| MachineryBuildingBlocks | 0:HasAddIn | True | Monitoring |
| MachineryBuildingBlocks | 0:HasAddIn | True | |  | | --- | | Monitoring | | Status | | MachineryItemState | |

The components of the *ConveyorBrakeType* have additional subcomponents which are defined in Table 26.

Table – ConveyorBrakeType Additional Subcomponents

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Source Path** | **Reference** | **NodeClass** | **BrowseName** | **DataType** | **TypeDefinition** | **Others** |
| |  | | --- | | Monitoring | | Status | | 0:HasComponent | Object | MachineryItemState |  | 4:MachineryItemState\_StateMachineType |  |
| |  | | --- | | Monitoring | | Status | | 0:HasProperty | Variable | BrakeReleased | 0:Boolean | 0:PropertyType |  |
| |  | | --- | | Monitoring | | Process | | 0:HasProperty | Variable | ReleaseTime | 0:Duration | 0:PropertyType |  |
| |  | | --- | | Monitoring | | Process | | 0:HasComponent | Variable | BreakingTorque | 0:Double | 0:AnalogUnitType |  |

*MachineryItemState* is used as defined in OPC 40001-1. It represents a StateMachine that shows the current machine state.

*BrakeReleased* is

*ReleaseTime* is

*BreakingTorque* is

## BeltType ObjectType Definition

### Overview

The *BeltType* provides ... and is formally defined in Table 27.

Table – BeltType Definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute** | **Value** | | | | |
| BrowseName | BeltType | | | | |
| IsAbstract | False | | | | |
| **References** | **Node Class** | **BrowseName** | **DataType** | **TypeDefinition** | **Other** |
| Subtype of the *5:MiningEquipmentType* defined in OPC 40560 | | | | | |
| 0:HasAddIn | Object | MiningEquipmentIdentification |  | 5:MiningEquipmentIdentificationType |  |
| 0:HasComponent | Object | Components |  | 4:MachineComponentsType |  |
| 0:HasComponent | Object | Monitoring |  | 4:MonitoringType |  |
| 0:HasComponent | Object | MachineProperties |  | 0:FolderType |  |
| 0:HasComponent | Object | MachineryBuildingBlocks |  | 0:FolderType |  |
| **Conformance Units** | | | | | |
|  | | | | | |
|  | | | | | |

*MiningEquipmentIdentification* represents numerous identification features of a mining equipment. It is a subtype of the *4:MachineIdentificationType*.

*Components* represents a collection of all identifiable components contained below this *Object*.

*Monitoring* represents a collection of *ObjectTypes* and *VariableTypes*, representing the current state of the process, that are not assigned to a component but to this *Object*.

*MachineProperties* is representing a collection of static information that are assigned to this *Object.*

*MachineryBuidlingBlocks* is representing a folder that directly references all those building blocks of the OPC UA for Machinery (OPC 40001-1, OPC 40001-3) which are implemented as an add-in.

The components of the *BeltType* have additional references which are defined in Table 28.

Table – BeltType Additional References

|  |  |  |  |
| --- | --- | --- | --- |
| **SourceBrowsePath** | **Reference Type** | **Is Forward** | **TargetBrowsePath** |
| MachineryBuildingBlocks | 0:HasAddIn | True | MiningEquipmentIdentification |
| MachineryBuildingBlocks | 0:HasAddIn | True | Components |
| MachineryBuildingBlocks | 0:HasAddIn | True | Monitoring |

The components of the *BeltType* have additional subcomponents which are defined in Table 29.

Table – BeltType Additional Subcomponents

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Source Path** | **Reference** | **NodeClass** | **BrowseName** | **DataType** | **TypeDefinition** | **Others** |
| Components | 0:HasComponent | Object | <BeltSection> |  | BeltSectionType |  |
| Components | 0:HasComponent | Object | <BeltSplice> |  | BeltSpliceType |  |
| |  | | --- | | Monitoring | | Status | | 0:HasComponent | Variable | BeltPosition | 0:Double | 0:AnalogUnitType |  |
| |  | | --- | | Monitoring | | Status | | 0:HasProperty | Variable | BeltSideDeflectionLeft | 0:Boolean | 0:PropertyType |  |
| |  | | --- | | Monitoring | | Status | | 0:HasProperty | Variable | BeltSideDeflectionRight | 0:Boolean | 0:PropertyType |  |
| |  | | --- | | Monitoring | | Process | | 0:HasComponent | Variable | BeltTension | 0:Double | 0:AnalogUnitType |  |
| |  | | --- | | Monitoring | | Process | | 0:HasComponent | Variable | BeltPreTension | 0:Double | 0:AnalogUnitType |  |
| MachineProperties | 0:HasComponent | Variable | BeltWidth | 0:Double | 0:AnalogUnitType |  |

*BeltSection* is

*BeltSplice* is

*BeltPosition* is

*BeltSideDeflectionLeft* is

*BeltSideDeflectionRight* is

*BeltTension* is

*BeltPreTension* is

*BeltWidth* is

## BeltSectionType ObjectType Definition

### Overview

The *BeltSectionType* provides ... and is formally defined in Table 30.

Table – BeltSectionType Definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute** | **Value** | | | | |
| BrowseName | BeltSectionType | | | | |
| IsAbstract | False | | | | |
| **References** | **Node Class** | **BrowseName** | **DataType** | **TypeDefinition** | **Other** |
| Subtype of the *0:BaseObjectType* defined in OPC 10000-5 | | | | | |
| 0:HasAddIn | Object | MiningEquipmentIdentification |  | 5:MiningEquipmentIdentificationType |  |
| 0:HasProperty | Variable | ProductionSite | 0:String | 0:PropertyType |  |
| 0:HasComponent | Variable | BeltSection | 0:Int32 | 0:AnalogUnitType |  |
| 0:HasProperty | Variable | EmbeddedSensorTag | 0:String | 0:PropertyType |  |
| 0:HasComponent | Variable | BeltThickness | 0:Double | 0:AnalogUnitType |  |
| 0:HasProperty | Variable | ConnectedBeltSplices | 0:Int32[] | 0: PropertyType |  |
| 0:HasComponent | Object | MachineryBuildingBlocks |  | 0:FolderType |  |
| **Conformance Units** | | | | | |
|  | | | | | |
|  | | | | | |

*MiningEquipmentIdentification* represents numerous identification features of a mining equipment. It is a subtype of the *4:MachineIdentificationType*.

*ProductionSite* is

*BeltSection* is

*EmbeddedSensorTag* is

*BeltThickness* is

*MachineryBuidlingBlocks* is representing a folder that directly references all those building blocks of the OPC UA for Machinery (OPC 40001-1, OPC 40001-3) which are implemented as an add-in.

The components of the *BeltSectionType* have additional references which are defined in Table 31.

Table – BeltSectionType Additional References

|  |  |  |  |
| --- | --- | --- | --- |
| **SourceBrowsePath** | **Reference Type** | **Is Forward** | **TargetBrowsePath** |
| MachineryBuildingBlocks | 0:HasAddIn | True | MiningEquipmentIdentification |

The components of the *BeltSectionType* have additional subcomponents which are defined in Table 32.

Table – BeltSectionType Additional Subcomponents

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Source Path** | **Reference** | **NodeClass** | **BrowseName** | **DataType** | **TypeDefinition** | **Others** |
| BeltThickness | 0:HasProperty | Variable | DateOfMeasurement | 0:DateTime | 0:PropertyType |  |

## BeltSpliceType ObjectType Definition

### Overview

The *BeltSpliceType* provides ... and is formally defined in Table 33.

Table – BeltSpliceType Definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute** | **Value** | | | | |
| BrowseName | BeltSpliceType | | | | |
| IsAbstract | False | | | | |
| **References** | **Node Class** | **BrowseName** | **DataType** | **TypeDefinition** | **Other** |
| Subtype of the *0:BaseObjectType* defined in OPC 10000-5 | | | | | |
| 0:HasAddIn | Object | MiningEquipmentIdentification |  | 5:MiningEquipmentIdentificationType |  |
| 0:HasComponent | Variable | Length | 0:Double | 0:AnalogUnitType |  |
| 0:HasComponent | Variable | NumberOfSteps | 0:Int32 | 0:AnalogUnitType |  |
| 0:HasComponent | Variable | BeltSpliceNumber | 0:Int32 | 0:AnalogUnitType |  |
| 0:HasComponent | Variable | StatusSplice |  |  |  |
| 0:HasComponent | Object | MachineryBuildingBlocks |  | 0:FolderType |  |
| **Conformance Units** | | | | | |
|  | | | | | |
|  | | | | | |

*MiningEquipmentIdentification* represents numerous identification features of a mining equipment. It is a subtype of the *4:MachineIdentificationType*.

*Length* is

*NumberOfSteps* is

*BeltSpliceNumber* is

*StatusSplice* is

*MachineryBuidlingBlocks* is representing a folder that directly references all those building blocks of the OPC UA for Machinery (OPC 40001-1, OPC 40001-3) which are implemented as an add-in.

The components of the *BeltSpliceType* have additional references which are defined in Table 34.

Table – BeltSpliceType Additional References

|  |  |  |  |
| --- | --- | --- | --- |
| **SourceBrowsePath** | **Reference Type** | **Is Forward** | **TargetBrowsePath** |
| MachineryBuildingBlocks | 0:HasAddIn | True | MiningEquipmentIdentification |

## PulleyType ObjectType Definition

### Overview

The *PulleyType* provides ... and is formally defined in Table 36.

Table – PulleyType Definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute** | **Value** | | | | |
| BrowseName | PulleyType | | | | |
| IsAbstract | False | | | | |
| **References** | **Node Class** | **BrowseName** | **DataType** | **TypeDefinition** | **Other** |
| Subtype of the *5:MiningEquipmentType* defined in OPC 40560 | | | | | |
| 0:HasAddIn | Object | MiningEquipmentIdentification |  | 5:MiningEquipmentIdentificationType |  |
| 0:HasComponent | Object | Monitoring |  | 4:MonitoringType |  |
| 0:HasComponent | Object | MachineProperties |  | 0:FolderType |  |
| 0:HasComponent | Object | MachineryBuildingBlocks |  | 0:FolderType |  |
| **Conformance Units** | | | | | |
|  | | | | | |
|  | | | | | |

*MiningEquipmentIdentification* represents numerous identification features of a mining equipment. It is a subtype of the *4:MachineIdentificationType*.

*Monitoring* represents a collection of *ObjectTypes* and *VariableTypes*, representing the current state of the process, that are not assigned to a component but to this *Object*.

*MachineProperties* is representing a collection of static information that are assigned to this *Object.*

*MachineryBuidlingBlocks* is representing a folder that directly references all those building blocks of the OPC UA for Machinery (OPC 40001-1, OPC 40001-3) which are implemented as an add-in.

The components of the *PulleyType* have additional references which are defined in Table 37.

Table – PulleyType Additional References

|  |  |  |  |
| --- | --- | --- | --- |
| **SourceBrowsePath** | **Reference Type** | **Is Forward** | **TargetBrowsePath** |
| MachineryBuildingBlocks | 0:HasAddIn | True | MiningEquipmentIdentification |
| MachineryBuildingBlocks | 0:HasAddIn | True | Monitoring |
| MachineryBuildingBlocks | 0:HasAddIn | True | |  | | --- | | Monitoring | | Status | | MachineryItemState | |

The components of the *PulleyType* have additional subcomponents which are defined in Table 38.

Table – PulleyType Additional Subcomponents

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Source Path** | **Reference** | **NodeClass** | **BrowseName** | **DataType** | **TypeDefinition** | **Others** |
| |  | | --- | | Monitoring | | Status | | 0:HasComponent | Object | MachineryItemState |  | 4:MachineryItemState\_StateMachineType |  |
| |  | | --- | | Monitoring | | Status | | 0:HasComponent | Variable | PullyEnumeration |  |  |  |
| |  | | --- | | Monitoring | | Process | | 0:HasComponent | Variable | <BearingTemperature> | 0:Double | 0:AnalogUnitType |  |
| |  | | --- | | Monitoring | | Process | | 0:HasComponent | Variable | RotationalSpeed | 0:Double | 0:AnalogUnitType |  |
| |  | | --- | | Monitoring | | Process | | 0:HasComponent | Variable | Vibrations | 0:Double | 0:AnalogUnitType |  |
| MachineProperties | 0:HasComponent | Variable | DiameterWithLagging | 0:Double | 0:AnalogUnitType |  |
| MachineProperties | 0:HasComponent | Variable | FaceWidth | 0:Double | 0:AnalogUnitType |  |

*MachineryItemState* is used as defined in OPC 40001-1. It represents a StateMachine that shows the current machine state.

*PulleyEnumeration* is

*BearingTemperature* is

*RotationalSpeed* is

*Vibrations* is

*DiameterWithLagging* is

*FaceWidth* is

## WeatherStationType ObjectType Definition

### Overview

The *WeatherStationType* provides ... and is formally defined in Table 37.

Table – WeatherStationType Definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute** | **Value** | | | | |
| BrowseName | WeatherStationType | | | | |
| IsAbstract | False | | | | |
| **References** | **Node Class** | **BrowseName** | **DataType** | **TypeDefinition** | **Other** |
| Subtype of the *5:MiningEquipmentType* defined in OPC 40560 | | | | | |
| 0:HasAddIn | Object | MiningEquipmentIdentification |  | 5:MiningEquipmentIdentificationType |  |
| 0:HasComponent | Object | Monitoring |  | 4:MonitoringType |  |
| 0:HasComponent | Object | MachineryBuildingBlocks |  | 0:FolderType |  |
| **Conformance Units** | | | | | |
|  | | | | | |
|  | | | | | |

*MiningEquipmentIdentification* represents numerous identification features of a mining equipment. It is a subtype of the *4:MachineIdentificationType*.

*Monitoring* represents a collection of *ObjectTypes* and *VariableTypes*, representing the current state of the process, that are not assigned to a component but to this *Object*.

*MachineryBuidlingBlocks* is representing a folder that directly references all those building blocks of the OPC UA for Machinery (OPC 40001-1, OPC 40001-3) which are implemented as an add-in.

The components of the *WeatherStationType* have additional references which are defined in Table 38.

Table – WeatherStationType Additional References

|  |  |  |  |
| --- | --- | --- | --- |
| **SourceBrowsePath** | **Reference Type** | **Is Forward** | **TargetBrowsePath** |
| MachineryBuildingBlocks | 0:HasAddIn | True | MiningEquipmentIdentification |
| MachineryBuildingBlocks | 0:HasAddIn | True | Monitoring |
| MachineryBuildingBlocks | 0:HasAddIn | True | |  | | --- | | Monitoring | | Status | | MachineryItemState | |

The components of the *WeatherStationType* have additional subcomponents which are defined in Table 39.

Table – WeatherStationType Additional Subcomponents

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Source Path** | **Reference** | **NodeClass** | **BrowseName** | **DataType** | **TypeDefinition** | **Others** |
| |  | | --- | | Monitoring | | Status | | 0:HasComponent | Object | MachineryItemState |  | 4:MachineryItemState\_StateMachineType |  |
| |  | | --- | | Monitoring | | Process | | 0:HasComponent | Variable | Temperature | 0:Double | 0:AnalogUnitType |  |
| |  | | --- | | Monitoring | | Process | | 0:HasComponent | Variable | Humidity | 0:Double | 0:AnalogUnitType |  |
| |  | | --- | | Monitoring | | Process | | 0:HasComponent | Variable | Percipitation | 0:Double | 0:AnalogUnitType |  |

*MachineryItemState* is used as defined in OPC 40001-1. It represents a StateMachine that shows the current machine state.

*Temperature* is

*Humidity* is

*Precipitation* is

## ScraperType ObjectType Definition

### Overview

The *ScraperType* provides ... and is formally defined in Table 40.

Table – ScraperType Definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute** | **Value** | | | | |
| BrowseName | ScraperType | | | | |
| IsAbstract | False | | | | |
| **References** | **Node Class** | **BrowseName** | **DataType** | **TypeDefinition** | **Other** |
| Subtype of the *5:MiningEquipmentType* defined in OPC 40560 | | | | | |
| 0:HasAddIn | Object | MiningEquipmentIdentification |  | 5:MiningEquipmentIdentificationType |  |
| 0:HasComponent | Object | Monitoring |  | 4:MonitoringType |  |
| 0:HasComponent | Object | MachineryBuildingBlocks |  | 0:FolderType |  |
| **Conformance Units** | | | | | |
|  | | | | | |
|  | | | | | |

*MiningEquipmentIdentification* represents numerous identification features of a mining equipment. It is a subtype of the *4:MachineIdentificationType*.

*Monitoring* represents a collection of *ObjectTypes* and *VariableTypes*, representing the current state of the process, that are not assigned to a component but to this *Object*.

*MachineryBuidlingBlocks* is representing a folder that directly references all those building blocks of the OPC UA for Machinery (OPC 40001-1, OPC 40001-3) which are implemented as an add-in.

The components of the *ScraperType* have additional references which are defined in Table 41.

Table – ScraperType Additional References

|  |  |  |  |
| --- | --- | --- | --- |
| **SourceBrowsePath** | **Reference Type** | **Is Forward** | **TargetBrowsePath** |
| MachineryBuildingBlocks | 0:HasAddIn | True | MiningEquipmentIdentification |
| MachineryBuildingBlocks | 0:HasAddIn | True | Monitoring |
| MachineryBuildingBlocks | 0:HasAddIn | True | |  | | --- | | Monitoring | | Status | | MachineryItemState | |

The components of the *ScraperType* have additional subcomponents which are defined in Table 42.

Table – ScraperType Additional Subcomponents

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Source Path** | **Reference** | **NodeClass** | **BrowseName** | **DataType** | **TypeDefinition** | **Others** |
| |  | | --- | | Monitoring | | Status | | 0:HasComponent | Object | MachineryItemState |  | 4:MachineryItemState\_StateMachineType |  |
| |  | | --- | | Monitoring | | Status | | 0:HasProperty | Variable | ScraperActive | 0:Boolean | 0:PropertyType |  |
| |  | | --- | | Monitoring | | Process | |  |  | CleanlinessOfBelt |  |  |  |

*MachineryItemState* is used as defined in OPC 40001-1. It represents a StateMachine that shows the current machine state.

*ScraperActive* is

## ConveyorRollType ObjectType Definition

### Overview

The *ConveyorRollType* provides ... and is formally defined in Table 52.

Table – ConveyorRollType Definition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute** | **Value** | | | | |
| BrowseName | ConveyorRollType | | | | |
| IsAbstract | False | | | | |
| **References** | **Node Class** | **BrowseName** | **DataType** | **TypeDefinition** | **Other** |
| Subtype of the *0:BaseObjectType* defined in OPC 10000-5 | | | | | |
| 0:HasProperty | Variable | ConveyorRollID | 0:String | 0:PropertyType |  |
| 0:HasComponent | Variable | <BearingTemperature> | 0:Double | 0:AnalogUnitType |  |
| 0:HasComponent | Variable | <BearingRotation> | 0:Double | 0:AnalogUnitType |  |
| 0:HasComponent | Variable | <Vibration> | 0:Double | 0:AnalogUnitType |  |
|  | Variable | <SensorNodeID> |  |  |  |
| **Conformance Units** | | | | | |
|  | | | | | |
|  | | | | | |

*ConveyorRollID* is

*BearingTemperature* is

*BearingRotation* is

*Vibration* is

*SensorNodeID* is

# Profiles and ConformanceUnits

xyz

# Namespaces

## Namespace Metadata

Table 41 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 53 – NamespaceMetadata Object for this Document

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Value** | | |
| BrowseName | http://opcfoundation.org/UA/<short name>/ | | |
| **Property** | | **DataType** | **Value** |
| NamespaceUri | | String | http://opcfoundation.org/UA/<short name>/ |
| NamespaceVersion | | String | 1.0 |
| NamespacePublicationDate | | DateTime | 2021-11-23 |
| 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 42 provides a list of mandatory and optional namespaces used in an <title> OPC UA *Server*.

Table 54 – 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 namespace shall have namespace index 1. | Mandatory |
| http://opcfoundation.org/UA/DI/ | Namespace for *NodeIds* and *BrowseNames* defined in OPC 10000-100. The namespace index is *Server* specific. | Mandatory |
| http://opcfoundation.org/UA/Machinery/ | Namespace for *NodeIds* and *BrowseNames* defined in OPC UA for Machinery (**Error! Reference source not found.**). The namespace index is *Server* specific. | Mandatory |
| http://opcfoundation.org/UA/<short name>/ | Namespace for *NodeIds* and *BrowseNames* defined in this document. The namespace index is *Server* specific. | Mandatory |
| Vendor specific types | A *Server* may provide vendor-specific types like types derived from *ObjectTypes* defined in this document in a vendor-specific namespace. | Optional |
| Vendor specific instances | A *Server* provides vendor-specific instances of the standard types or vendor-specific instances of vendor-specific types in a vendor-specific namespace.  It is recommended to separate vendor specific types and vendor specific instances into two or more namespaces. | Mandatory |

Table 43 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 55 – 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/IA/ | 3 | 3:BasicStacklightType |
| http://opcfoundation.org/UA/Machinery/ | 4 | 4:MachineIdentificaionType |
| http://opcfoundation.org/UA/Mining/General/ | 5 | 5:MiningEquipmentType |

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](file:///D:/opc/UA/Companion/Template/http)[://opcfoundation.org/UA/<short name>/](http://opcfoundation.org/UA/POWERLINK/)

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

Where <short name> is described in **Error! Reference source not found.** 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>/](https://reference.opcfoundation.org/nodesets?u=http://opcfoundation.org/UA/%3cshort%20name%3e/)

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 **Error! Reference source not found.**). 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 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.

* 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 13 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>

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