OPC 30070-2

OPC UA for MTConnect®

Part 2: Asset Model

Release 2.00.00 RC1

June 6, 2019

OPC UA Companion Specification

Specification Type:	Industry Standard Specification	Comments:	
Document Number	**OPC 30070-2**		
Title:	OPC UA for MTConnect® Part 2: Asset Model	Date:	June 6, 2019
Version:	Release 2.00.00 RC1	Software:	LaTeX
Authors:	William Sobel, Randy Armstrong, John Turner, Russell Waddell, Shaurabh Singh	Source:	OPC_UA_MTConnect_2.00.00 RC1.pdf
Owner:	MTConnect Institute	Status:	Release

Document History

Version	Date	Reason	Comments	Mantis
2.00.00 RC1	2019-06-06	Initial	Initial Release Candidate	

Contents

List of Figures

List of Tables

OPC Foundation and MTConnect® Institute

AGREEMENT OF USE

conformance with this specification.

TRADEMARKS

MTConnect® is a registered trademark of the The Association for Manufacturing Technology (AMT).

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.

1 Scope

In September 2010, the OPC Foundation and the MTConnect Institute signed a memorandum of understanding to extend the reach of existing manufacturing data exchange standards to:

- Evolve the existing standards from each organization to provide complete manufacturing technology interoperability.
- Provide a mechanism for continuous improvement of those standards and specifications.
- Support the evolution of digital manufacturing systems.
- Provide a coordinating function to harmonize work between the organizations.
- Educate customers and suppliers on the standards and specifications.
- Provide a foundation for adopting the standards, specifications, and associated technology into real products.

The first document produced was the MTConnect-OPC UA Companion Specification, Version 1.0 (2012), which defines a method for interoperability between the standards. It also identifies how the standards can be used together in manufacturing systems.

This document, OPC Unified Architecture for MTConnect Companion Specification, Version 2.0, updates the original companion specification and incorporates the latest capabilities and functions.

The technologies provided from these two organizations include:

<u>OPC Foundation</u>

OPC is an 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

MTConnect Institute

MTConnect is a data and information exchange standard that is based on a data dictionary of terms describing information associated with manufacturing operations. The standard also defines a series of semantic data models that provide a clear and unambiguous representation of how that information relates to a manufacturing operation. The MTConnect Standard has been designed to:

- Enhance the data acquisition capabilities from equipment in manufacturing facilities
- Expand the use of data driven decision making in manufacturing operations
- Enable software applications and manufacturing equipment to move toward a plug-and-play environment to reduce the cost of integration of manufacturing software systems.

The MTConnect Institute is responsible for development of the MTConnect Standard. The Institute is a 501(c)(6) not-for-profit standards development organization and is a subsidiary of The Association for Manufacturing Technology (AMT). Its mission is to create open standards to foster greater interoperability between devices and clients by defining the structure and terminology used in communications in the discrete parts manufacturing sector.

2 OPC Unified Architecture for MTConnect Companion Specification Goals

This companion specification has been develop to provide the following:

- Encourage broad and rapid adoption. Support suppliers of equipment and software in adopting the included technology.
- Allow both standards to evolve independently without jeopardizing backwards compatibility with previous implementations.
- Support extensibility to allow customized machine or installation specific use cases without jeopardizing compatibility.
- Maintain the non-proprietary philosophy of each standards body in support of increased productivity in manufacturing.

3 Normative References

The following referenced documents are indispensable for applying this specification. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

3.1 OPC UA References

[UA Part 01] *OPC UA Specification: Part 1 – Overview and Concepts*. URL: http://www.opcfoundation.org/UA/Part1/.

[UA Part 02] *OPC UA Specification: Part 2 – Security Model*. URL: http://www.opcfoundation.org/UA/Part2/.

[UA Part 03] *OPC UA Specification: Part 3 – Address Space Model*. URL: http://www.opcfoundation.org/UA/Part3/.

[UA Part 04] *OPC UA Specification: Part 4 – Services*. URL: http://www.opcfoundation.org/UA/Part4/.

[UA Part 05] *OPC UA Specification: Part 5 – Information Model*. URL: http://www.opcfoundation.org/UA/Part5/.

[UA Part 06] *OPC UA Specification: Part 6 – Mappings*. URL: http://www.opcfoundation.org/UA/Part6/.

[UA Part 07] *OPC UA Specification: Part 7 – Profiles*. URL: http://www.opcfoundation.org/UA/Part7/.

[UA Part 08] *OPC UA Specification: Part 8 – Data Access*. URL: http://www.opcfoundation.org/UA/Part8/.

[UA Part 09] *OPC UA Specification: Part 9 – Alarms and Conditions*. URL: http://www.opcfoundation.org/UA/Part9/.

[UA Part 10] *OPC UA Specification: Part 10 – Programs*. URL: http://www.opcfoundation.org/UA/Part10/.

3.2 MTConnect References

[MTConnect Part 1.0] *MTConnect Standard Part 1.0 - Overview and Fundamentals*. Version 1.4.0. URL: http://bit.ly/2CaOlt1.

[MTConnect Part 2.0] *MTConnect Standard: Part 2.0 - Devices Information Model*. Version 1.4.0. URL: http://bit.ly/2OVfEy6.

[MTConnect Part 3.0] *MTConnect Standard: Part 3.0 - Streams Information Model*. Version 1.4.0. URL: http://bit.ly/2pMob8q.

[MTConnect Part 4.0] *MTConnect Standard: Part 4.0 - Assets Information Model*. Version 1.4.0. URL: http://bit.ly/2yyLc2D.

[MTConnect Part 4.1] *MTConnect Standard: Part 4.1 - Cutting Tools*. Version 1.4.0. URL: http://bit.ly/2A5eDeN.

[MTConnect Part 5.0] *MTConnect Standard: Part 5.0 - Interfaces*. Version 1.4.0. URL: http://bit.ly/2pPNGFY.

4 Terms, Definitions and Conventions

4.1 Overview

The basic concepts of OPC UA and MTConnect are pre-requisites for understanding and interpreting the content provided in this companion specification. Additionally, the terms and definitions given in [UA Part 01], [UA Part 02], [UA Part 03], [UA Part 05], [UA Part 07], [UA Part 10], and [MTConnect Part 1.0], (see section 3), as well as the following, apply to this document.

4.2 Conventions

Following are basic conventions that shall be followed for all formal definitions used: MTConnect Terms will be displayed as follows using italic font (*MTConnect Term*). OPC UA Terms will use bold italic fonts (***OPC UA Term***). Terms will be linked to the associated glossary entry if available.

MTConnect Extensible Markup Language (XML) literals and code will appear in monospace `MTConnectCode` and OPC UA literals and UA Model will appear as bold monospace **`UAObjectsAndTypes`**.

4.3 Terms and Acronnyms

4.3.1 Conventions for Node descriptions

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

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

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

- If the ***TargetNode*** is a component of the ***Node*** being defined in the table the ***Attributes*** of the composed Node are defined in the same row of the table.
- The **DataType** is only specified for Variables; "[<number>]" indicates a single-dimensional array, for multi-dimensional arrays the expression is repeated for each dimension (e.g. [2][3] for a two-dimensional array). For all arrays the ***ArrayDimensions*** is set as identified by **<number>** values. If no **<number>** is set, the corresponding dimension is set to 0, indicating an unknown size. If no number is provided at all the ***ArrayDimensions*** can be omitted. If no brackets are provided, it identifies a scalar **DataType** and the **ValueRank** is set to the corresponding value (see [UA Part 03]). 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 [UA Part 03]) and the ***ArrayDimensions*** is set to **null** or is omitted. Examples are given in Table 1.
- The ***Type Definition*** is specified for ***Objects*** and ***Variables***.
- The ***Type Definition*** 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 **ModellingRule**. In the ***AddressSpace***, the ***Node*** shall use a **HasModellingRule** ***Reference*** to point to the corresponding **ModellingRule** ***Object***.

Table 1: Examples of DataTypes

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

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

Nodes of all other ***NodeClasses*** cannot be defined in the same table; therefore only the used **ReferenceType**, their ***NodeClass*** and their **BrowseName** are specified. A reference to another part of this document points to their definition.

Table 2 illustrates the table. If no components are provided, the **DataType**, ***Type Definition*** and **ModellingRule** columns may be omitted and only a Comment column is introduced to point to the ***Node*** definition.

Table 2: Type Definition Table

Attribute	**Value**				
Attribute name	Attribute value. If it is an optional Attribute that is not set "–" will be used.				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modeling Rule**
ReferenceType name	NodeClass of the target Node.	BrowseName of the target Node. If the Reference is to be instantiated by the server, then the value of the target Node's BrowseName is "–".	DataType of the referenced Node, only applicable for Variable.	TypeDefinition of the referenced Node, only applicable for Variable and Object.	Referenced ModellingRule of the referenced Object.
Note: Notes referencing footnotes of the table content.					

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

4.3.2 NodeIds and BrowseNames

4.3.2.1 NodeIds

The **`NodeIds`** of all ***Nodes*** described in this standard are only symbolic names. Annex A defines the actual **`NodeIds`**.

The symbolic name of each ***Node*** defined in this specification is its **`BrowseName`**, or, when it is part of another Node, the **`BrowseName`** of the other ***Node***, a ".", and the **`BrowseName`** of itself. In this case "part of" means that the whole has a **`HasProperty`** or **`HasComponent`** Reference to its part. Since all ***Nodes*** not being part of another ***Node*** have a unique name in this specification, the symbolic name is unique.

The namespace for all **`NodeIds`** defined in this specification is defined in Annex A. The namespace for this ***NamespaceIndex*** is Server-specific and depends on the position of the namespace URI in the server namespace table.

Note that this specification not only defines concrete ***Nodes***, but also requires that some Nodes shall be generated, for example one for each Session running on the Server. The **`NodeIds`** of those ***Nodes*** are Server-specific, including the namespace. But the ***NamespaceIndex*** of those ***Nodes*** cannot be the ***NamespaceIndex*** used for the Nodes defined in this specification, because they are not defined by this specification but generated by the

Server.

4.3.2.2 BrowseNames

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

4.3.3 Common Attributes

4.3.3.1 General

The ***Attributes*** of ***Nodes***, their ***DataTypes*** and descriptions are defined in [UA Part 03]. ***Attributes*** not marked as optional are mandatory and shall be provided by a Server. The following tables define if the ***Attribute*** value is defined by this specification or if it is server-specific.

For all Nodes specified in this specification, the ***Attributes*** named in Table 3 shall be set as specified in the table.

Table 3: Common Node Attributes

Attribute	Value
DisplayName	The DisplayName is a LocalizedText. Each server shall provide the DisplayName identical to the BrowseName of the Node for the LocaleId "en". Whether the server provides translated names for other LocaleIds is server-specific.
Description	Optionally a server-specific description is provided.
NodeClass	Shall reflect the NodeClass of the Node.
NodeId	The NodeId is described by BrowseNames.
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.

4.3.3.2 Objects

For all `Objects` specified in this specification, the `Attributes` named in Table 4 shall be set as specified in the Table 4. The definitions for the `Attributes` can be found in OPC [UA Part 03].

Table 4: Common Object Attributes

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

4.3.3.3 Variables

For all ***Variables*** specified in this specification, the ***Attributes*** named in Table 5 shall be set as specified in the table. The definitions for the ***Attributes*** can be found in [UA Part 03].

Table 5: Common Variable Attributes

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

4.3.3.4 VariableTypes

For all **`VariableType`** specified in this specification, the ***Attributes*** named in Table 6 shall be set as specified in the table. The definitions for the ***Attributes*** can be found in [UA Part 03].

Table 6: Common VariableTypes Attributes

Attribute	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.

4.3.3.5 Methods

For all **`Methods`** specified in this specification, the ***Attribute*** named in Table 7 shall be set as specified in the table. The definitions for the ***Attributes*** can be found in [UA Part 03].

Table 7: Common Method Attributes

Attribute	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.

5 Introduction to MTConnect and OPC UA

5.1 MTConnect

MTConnect is a data and information exchange standard based on a data dictionary of terms describing information associated with manufacturing operations. The standard also defines a series of semantic data models that provide a clear and unambiguous representation of how that information relates to a manufacturing operation. The MTConnect Standard has been designed to enhance the data acquisition capabilities from equipment in manufacturing facilities, expand the use of data-driven decision making in manufacturing operations, and enable software applications and manufacturing equipment to move toward a plug-and-play environment to reduce the cost of integration of manufacturing software systems. The MTConnect standard supports two primary communications methods - Request/Response and Publish/Subscribe. Although the MTConnect Standard has been defined for manufacturing, it can also be readily applied to other application areas.

The MTConnect Standard is an open, royalty free standard - meaning that it is available for anyone to download, implement, and utilize in software systems at no cost. The semantic data models defined in the MTConnect standard provide the information required to fully characterize data with both a clear and unambiguous meaning and a mechanism to directly relate that data to the manufacturing operation where the data originated. Without a semantic data model, client software applications must apply an additional layer of logic to convey as much meaning. The MTConnect modeling approach allows applications to easily interpret data from a wide variety of data sources, reducing complexity and development effort. Where the data dictionary and semantic data models are insufficient, MTConnect can be extended with additional data items and information models.

MTConnect is designed to maximize interoperability with other standards, applications, and manufacturing equipment, and uses a variety of other standards to do so. Examples in the standard are based on Hypertext Transfer Protocol (HTTP) for transport protocol and XML for representing semantic data models. The transport protocol and the programming language used to represent or transfer the information provided by the semantic data models are not restricted in the standard, although there is a minimum requirement to support HTTP Representational State Transfer (REST) protocol and XML. Other protocols and programming languages may be used to represent the semantic models and/or transport the information provided by these data models between an MTConnect *Agent* (server) and a client software application.

The standard was initially sponsored by AMT in 2008. AMT formed the MTConnect Institute in 2011 to further standard development and engage a wider community. The role of the Institute is to support the continued development of the standard and to expand the deployment of MTConnect compliant technologies throughout industry. The Institute has over 250 member companies world-wide.

5.1.1 Data Dictionary

The Data Dictionary defines a consistent set of terms that are used to describe information and data gathered from shop floor operations. When various pieces of equipment publish information using this common Data Dictionary, that data is easier to understand and can be used directly for further analysis without requiring additional manipulation to get the data into a common format. By utilizing the Data Dictionary, equipment can now publish data that this "self-describing" - meaning the data not only provides values, but also provides essential meaning for the data; including units, tag names, scaling information, and any other information that may be needed for a software application to fully understand both the meaning of the data and the relevance of that data to the manufacturing process.

The users of this data no longer have to define this information each place the data is used.

5.1.2 Semantic Data Models

The Semantic Data Models defined in the MTConnect Standard are used to further enhance the meaning of the information published from equipment. These models are used to represent the physical and logical configuration for a piece of equipment and the correlation between each piece of data and the part or function in the piece of equipment that the data is most closely related.

The data reported by the machines is defined and organized based on the semantic Data Models defined in the MTConnect Standard.

Historically, significant configuration work was required to qualify every piece of data collected by giving it an identity; scaling it to common units, when required; and to characterize that data with whatever additional information was necessary to define the full meaning of the data. This same process for qualifying the data had to be replicated for every software application that needed to use this data.

When data is available directly from shop floor equipment that is fully qualified with an identity and all the additional information that is needed to interpret that data, software applications can be deployed more quickly and at a lower cost.

Structured semantic data provides a solid foundation that allows software implementers to focus more of their energies and time on enhanced analysis and decision making instead of constantly manipulating raw data into a usable form - reducing the time and effort to deploy and maintain software systems.

In an MTConnect compliant system, configuration management of a data collection system is virtually eliminated since data definition and transformation occurs at the piece of equipment. When changes occur, those changes can be automatically detected by the client software application(s) since each piece of equipment can publish its current configuration containing all semantics and data types.

5.1.3 Fundamentals of MTConnect

The MTConnect Standard is built upon other communications and software standards that are already heavily used in manufacturing facilities - HTTP, Ethernet, and XML.

Pieces of equipment publish information to an MTConnect *Agent*. Software applications request information relating to a piece of equipment by making an HTTP Request. The *Agent* responds to that request by publishing a MTConnect Response Document which is a text document encoded using XML.

By leveraging already existing standards, implementers of software solutions utilizing MTConnect have immediate access to a maximum number of software tools for creating and deploying software solutions. This also positions MTConnect for the highest level of interoperability with other standards, software applications, and equipment used throughout manufacturing operations.

MTConnect is implemented as a read-only communications solution. This means that a software application can read information from a piece of equipment, but it cannot write information directly to that equipment or cause the equipment to perform any specific actions. This is especially relevant as industrial internet and cyber physical systems' safety issues become more important. The read-only feature also makes the MTConnect Standard easier for integrators to implement.

Many manufacturing activities require a piece of equipment to initiate a specific action or function based on decisions or information from other pieces of equipment, software systems, or human intervention. MTConnect addresses this scenario through the Interfaces Interaction Model (MTConnect Part 5.0 [MTConnect Part 5.0]). This interaction model defines a standard methodology for pieces of equipment to directly exchange information without any one piece of equipment writing data or instructions to the other piece of equipment. This interaction model is commonly referred to as Read-Read where one piece of equipment Requests an action or activity to be performed, and the other piece of equipment "Reads" this requirement and independently decides how and when to Respond to that Request.

The central component of every MTConnect System is an MTConnect *Agent*. The agent provides the critical link between a piece of equipment and client software applications. The *Agent* performs several tasks within an MTConnect System. The two major functions provided by the *Agent* are the collection, organization, and storage of data published from one or multiple pieces of equipment and to then respond to requests for this data from client software applications.

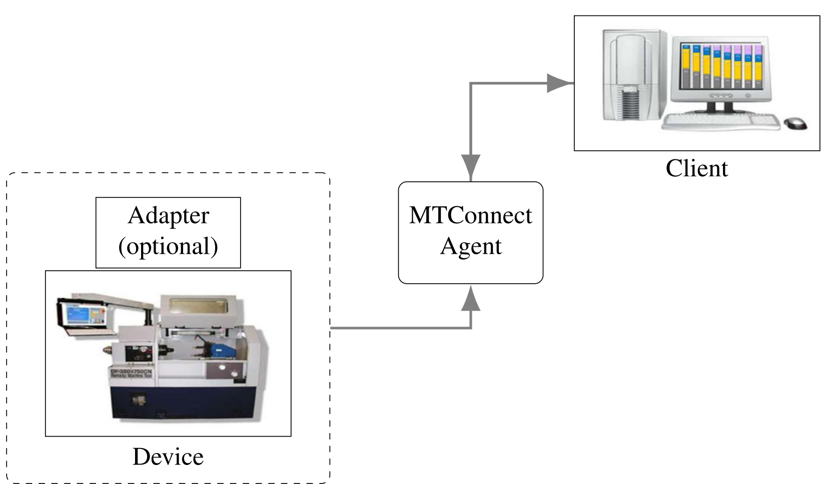


Figure 1: MTConnect Architecture Overview

In an MTConnect system, the term *"a piece of equipment"* can represent any intelligent data source that can produce data. Traditionally, a piece of equipment is thought of as a machine. However, a piece of equipment can also be a computer, an intelligent sensor system, a data base, and any number of other sources of data.

Some pieces of equipment require an Adapter which transforms data from its native form into MTConnect specific terms, and then publish that data to an MTConnect Agent.

5.2 Introduction to OPC Unified Architecture

OPC UA is an open and royalty free set of standards designed as a universal communications protocol. While there are numerous communication solutions available, OPC UA has key advantages:

- A state of art security model (see [UA Part 02]).
- A fault tolerant communication protocol.
- An information modeling 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 powerful semantic models such as MTConnect, 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 enterprise resource planning (ERP) systems. OPC UA devices process information locally and then provide that data in a consistent format to any application requesting data - ERP, manufacturing execution system (MES), Production Management System (PMS), Maintenance Systems, Human Machine Interface (HMI), Smartphone or a standard Browser, for examples. For a more complete overview see [UA Part 01].

5.2.1 Basics of OPC UA

As an Open Standard, OPC UA is based on standard Internet technologies - Transmission Control Protocol/Internet Protocol (TCP/IP), HTTP and Web Sockets.

As an Extensible Standard, OPC UA provides a set of services (see [UA Part 04]) 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 base thin clients and ERP systems. The full scope of OPC UA applications are shown in Figure 2.

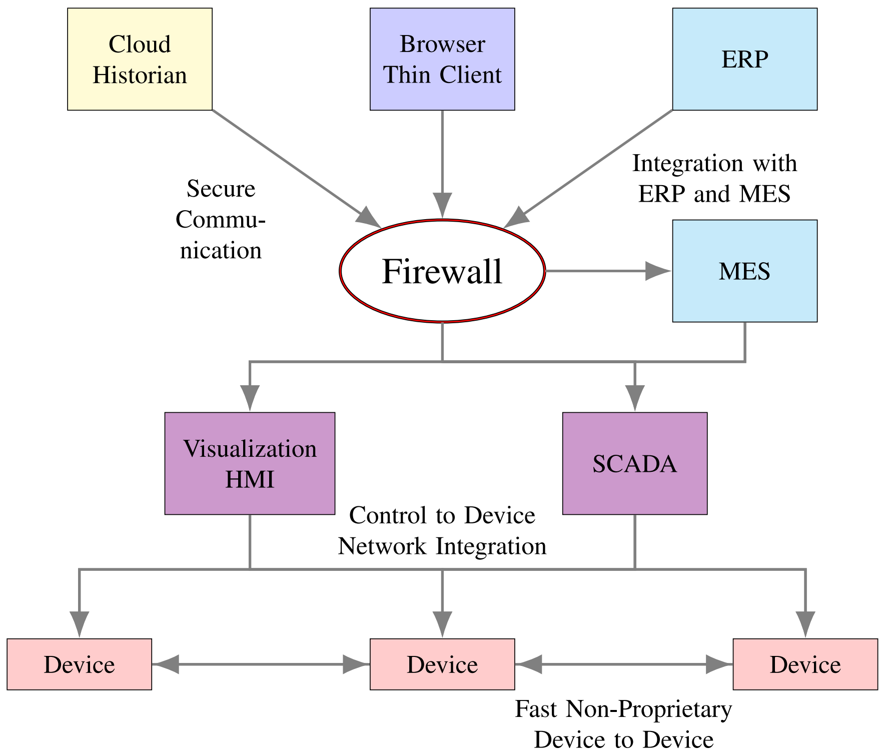


Figure 2: The Scope of OPC UA within an Enterprise

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.

5.2.2 Information Modeling in OPC UA

5.2.2.1 Concepts

OPC UA provides a framework that can be used to represent complex information as `Objects` in an `AddressSpace` which can be accessed with standard services. These

Objects consist of ***Nodes*** connected by `References`. Different classes of Nodes convey different semantics. For example, a ***Variable*** Node represents a value that can be read or written. The `Variable` Node has an associated **`DataType`** that can define the actual value, such as a string, float, structure etc. It can also describe the `Variable` value as a variant. A Method Node represents a function that can be called. Every Node has a number of `Attributes` including a unique identifier called a `NodeId` and non-localized name called as **`BrowseName`**. An `Object` representing a `Reservation` is shown in Figure 3.

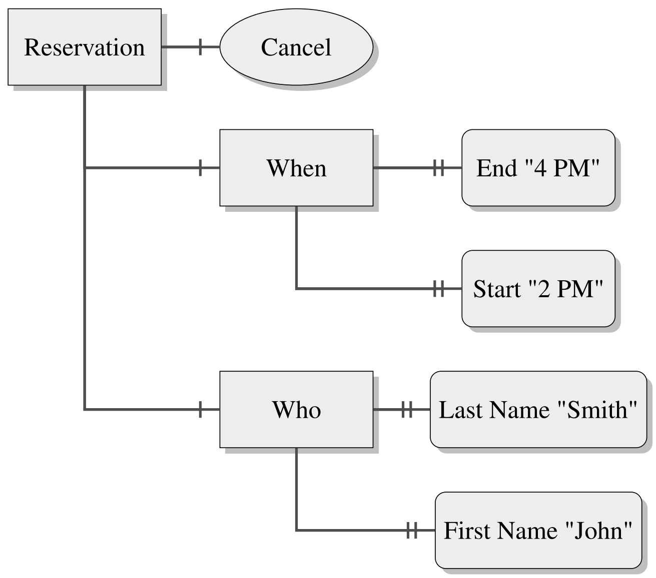


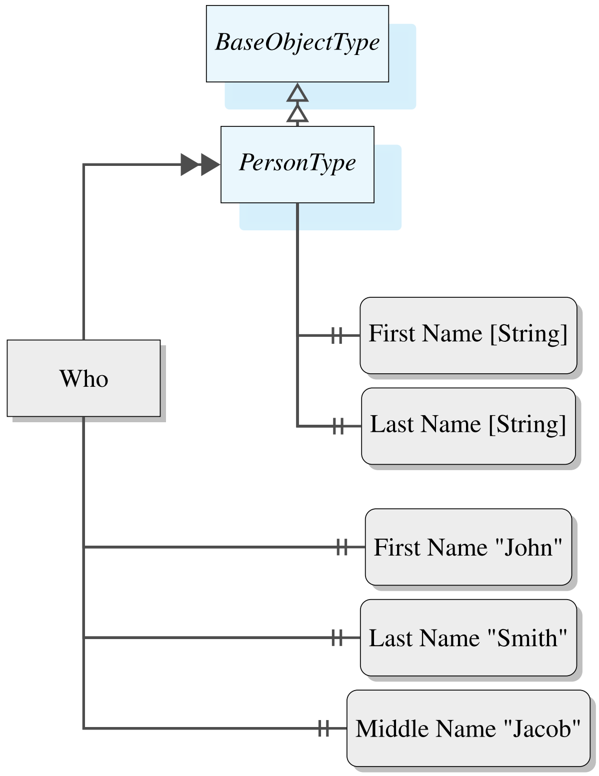
Figure 3: A Basic Object in an OPC UA Address Space

`Object` and `Variable Nodes` are called `Instance Nodes` and they always reference a Type Definition (`ObjectType` or `VariableType`) Node which describes their semantics and structure. Figure 4 illustrates the relationship between an Instance and its Type Definition.

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 4 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 child. 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 devices implement.

OPC UA also supports the concept of sub typing. This allows a modeler to take an existing Type and extend it. There are rules regarding sub typing defined in [UA Part 03], but in general they allow the additions to a given type or the restriction of a **`DataType`** to a more specific data type. For example the modeler may decide that the existing `ObjectType` in some cases needs an additional variable. The modeler can create a subtype of the

ObjectType and add the variable. A client that is expecting the parent type can treat the new ObjectType as if it was of the parent ObjectType and just ignore the additional variable. A client that understands the new subtype may display or otherwise process the additional variable. With regard to **DataTypes**, if a variable is defined to have a numeric value, a sub type could restrict the Value to a float.



An instance of PersonType represents a human

Figure 4: The Relationship between Type Definitions and Instances

References allow Nodes to be connected together 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 the existing ReferenceType. Subtypes inherit the semantics of the parent but may add additional restrictions. Figure 5 and Figure 6 depict several references connecting different Objects.

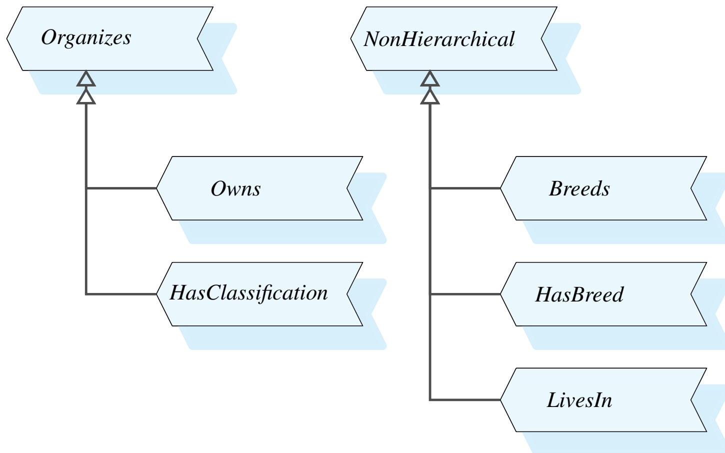


Figure 5: Reference Types from other Reference Types

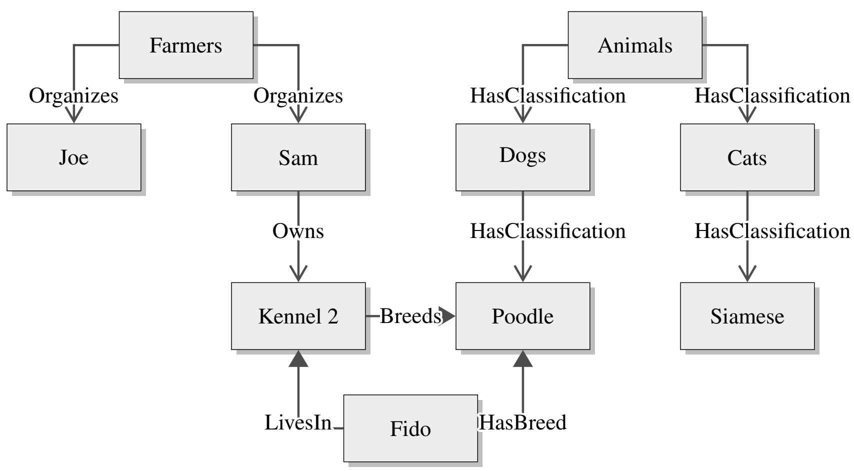


Figure 6: References showing hierarchies and/or relationships

The figures above use a notation that was developed for the OPC UA specification. The notation is summarized in Figure 7 and Figure 8 . 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 ***Node*** in the `AddressSpace` of an OPC UA Server.

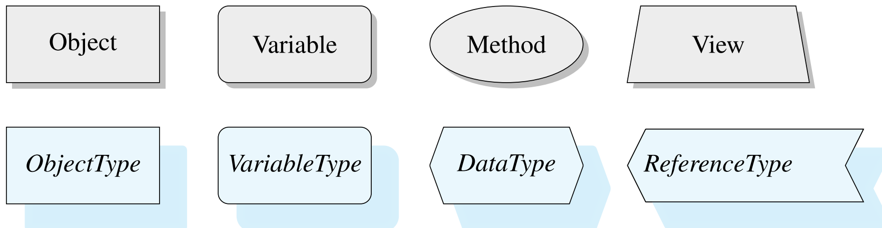


Figure 7: The OPC UA Information Model Notation for Types and Instances

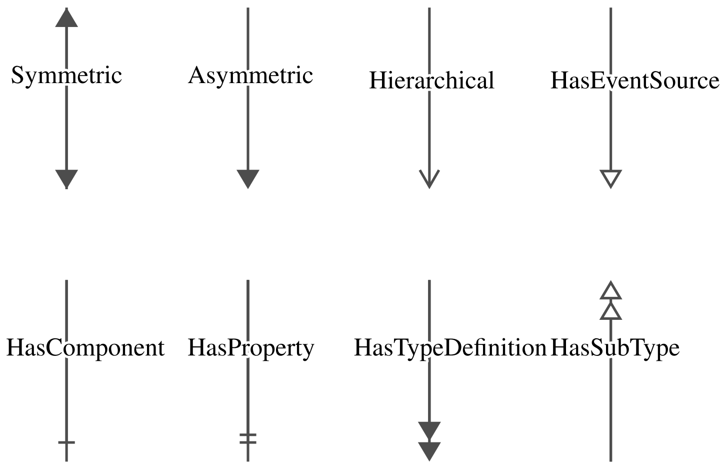


Figure 8: The OPC UA Standard References

A complete description of the different types of ***Nodes*** and `References` can be found in [UA Part 03] and the base structure is described in [UA Part 05]. OPC UA specification defines a very wide range of functionality in its basic information model. It is not expected that all clients or servers support all functionality in the OPC UA specifications. OPC UA includes the concept of profiles, which segment the functionality into testable certifiable units. This allows the development of companion specification (such as MTConnect-OPC UA) that can describe the subset of functionality that is expected to be implemented. The profiles do not restrict functionality, but generate requirements for a minimum set of functionality (see [UA Part 07])

5.2.2.2 Namespaces

OPC UA allows information from many different sources to be combined into a single coherent address space. `Namespaces` are used to make this possible by eliminating naming and id conflicts between information from different sources. `Namespaces` in OPC UA have a globally unique string called a `NamespaceUri` and a locally unique integer called a `NamespaceIndex`. The `NamespaceIndex` is only unique within the context of a Session between an OPC UA Client and an OPC UA Server. All of the web services defined for OPC UA use the `NamespaceIndex` to specify the `Namespace` for qualified values.

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

`QualifiedNames` are non-localized names qualified with a `Namespace`. They are used for the `BrowseNames` of Nodes and allow the same Names to be used by different information models without conflict. The `BrowseName` is used to identify the children within a `TypeDefinitions`. Instances of a `TypeDefinition` are expected to have children with the same `BrowseNames`. `TypeDefinitions` are not allowed to have children with duplicate `BrowseNames`; however, Instances do not have that restriction.

5.2.2.3 Companion Specifications

An OPC UA companion specification for an industry specific vertical market describes an Information Model by defining `ObjectTypes`, `VariableTypes`, `DataTypes` and `ReferenceTypes` (see section 4.2) that represent the concepts used in the vertical market, and potentially also well-defined Objects as entry points into the `AddressSpace`.

6 Mapping the MTConnect Information Model to OPC UA

This section describes a Unified Modeling Language (UML) representation of MTConnect semantic data models for mapping MTConnect into OLE for Process Control (OPC) Unified Architecture (UA). More detailed information is provided in Section 7 for each data type.

OPC UA defines abstractions representing data, relationships, and events from devices. The abstractions do not provide the semantic meaning; they provide a structure to convey the meta-data and the values as they change. The OPC UA model has the base building blocks to represent an ontological model where the specific ontology is povided by companion specification for a specific domain.

MTConnect has similar capabilities but uses a different structural model where the meta-data and the streaming values are in separate documents to normalize the data flow in a similar way that many publish-subscribe protocols separate the structure from the data. MTConnect also supports a store-and-forward capability like many message brokers in a Message Oriented Middleware (MOM) architecture to enable resilience and recovery of data in the event of connectivity problems.

When translating from MTConnect to OPC UA, the MTConnect abstractions of *DataItems* are converted using the OPC UA **`DataVariable`** abstractions as given in [UA Part 08]. The relationships are mapped to multiple **`DataVariable`** types where the category and the type determine the correct mapping. Conditions are mapped a sub-type of the OPC UA ***ConditionType*** in a similar way. The behavior of the OPC UA ***Conditions*** can be found in [UA Part 09].

6.1 MTConnect UML Representation of OPC

6.2 MTConnect Information Model

The MTConnect information model has the following abstractions:

1. *Components*
2. *DataItems*
3. *Configuration*
4. *Compositions*
5. *Assets*

The first concern of the MTConnect OPC UA companion specification is the *Device* model covered in MTConnect [MTConnect Part 2.0] and [MTConnect Part 3.0]. The top-level *Component* of any MTConnect information model is the *Device*. A *Component* represents a logical part or a collection of parts of a piece of equipment. The *DataItems* represent information that is communicated from *Components*, and the representation and communication of the information are covered in [MTConnect Part 3.0]. The *Compositions* are the lowest level of contextualization for MTConnect *Components* that do not have any structure but can be associated with *DataItems* to provide additional context.

The *Configuration* is a collection of information about the component that provides more detail about its capabilities. The standard has only specified the `SensorConfiguration` at this point.

Assets are complex information models that provide a point in time consistent set of information about the use of a physical or logical entity in the manufacturing process. These models, for example, may represent a cutting tool, a program, or a process. The *Assets* will be covered in a subsequent companion specification. The only assets currently in the MTConnect standard are `CuttingTool` and `CuttingToolArchetype`. Refer to MTConnect Part 4.0 [MTConnect Part 4.0] and MTConnect Part 4.1 [MTConnect Part 4.1].

The specification uses examples to illustrate the process of conversion from XML to OPC UA; the following sections cover the main points and concerns when converting an MTConnect Device model to a Nodeset. Following the metamodel discussion will be a section on the handling of streaming data and mapping to the correct data items.

6.3 Mapping The Model

7 MTConnect OPC UA Types

7.1 Assets

7.1.1 Defintion of `MTAssetType`

An Asset is a container type XML element used to organize information describing an entity that is not a piece of equipment. Asset is an abstract type and will never appear directly in the MTConnect information model.

Table 8: `MTAssetType` Definition

Attribute	Value				
BrowseName	MTAssetType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of BaseObjectType (See [UA Part 05] Documentation)					
HasSubtype	ObjectType	MTCuttingToolArchetypeType		See section 7.2.5	
HasSubtype	ObjectType	MTCuttingToolType		See section 7.2.10	
HasProperty	Variable	AssetId	String	PropertyType	Mandatory
HasProperty	Variable	DeviceUuid	String	PropertyType	Mandatory
HasProperty	Variable	MTDescription	String	PropertyType	Optional
HasProperty	Variable	Removed	Boolean	PropertyType	Optional
HasProperty	Variable	Timestamp	DateTime	PropertyType	Mandatory

7.2 Cutting Tool

7.2.1 Defintion of `MTCutterStatusType`

The values CutterStatus element can be a combined set of Status elements. The MTConnect Standard allows any set of statuses to be combined, but only certain combinations make sense. A Cutting Tool **SHOULD NOT** be both `NEW` and `USED` at the same time. There are no rules in the schema to enforce this, but this is left to the implementer. The following combinations **MUST NOT** occur:

- `NEW` **MUST NOT** be used with `USED`, `RECONDITIONED`, or `EXPIRED`.
- `UNKNOWN` **MUST NOT** be used with any other status.
- `ALLOCATED` and `UNALLOCATED` **MUST NOT** be used together.
- `AVAILABLE` and `UNAVAILABLE` **MUST NOT** be used together.

- If the tool is EXPIRED, BROKEN, or NOT_REGISTERED it **MUST NOT** be AVAILABLE.
- All other combinations are allowed.

Table 9: MTCutterStatusType Definition

Attribute	Value				
BrowseName	MTCutterStatusType				
IsAbstract	False				
ValueRank	-2				
DataType	UInteger				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modeling-Rule**
Subtype of MultiStateDiscreteType (See [UA Part 08] Documentation)					
HasComponent	Variable	EnumValues	CutterStatusDataType	CutterStatusDataType	Mandatory

7.2.1.1 Referenced Properties and Objects

- **Allowable Values** for CutterStatusDataType

Table 10: CutterStatusDataType Enumeration

Name	Index
AVAILABLE	1
ALLOCATED	2
BROKEN	4
EXPIRED	16
MEASURED	32
NEW	64
NOT_REGISTERED	128
RECONDITIONED	256
UNALLOCATE	512
UNAVAILABLE	1024
UNKNOWN	2048
USED	4096

7.2.2 Defintion of MTCuttingItemType

Table 11: `MTCuttingItemType` Definition

Attribute	Value				
BrowseName	MTCuttingItemType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modeling-Rule**
Subtype of BaseObjectType (See [UA Part 05] Documentation)					
HasProperty	Variable	Grade	String	PropertyType	Optional
HasProperty	Variable	Indices	String	PropertyType	Mandatory
HasProperty	Variable	ItemId	String	PropertyType	Optional
HasProperty	Variable	Locus	String	PropertyType	Optional
HasProperty	Variable	Manufactures	String[]	PropertyType	Optional
HasProperty	Variable	MTDescription	String	PropertyType	Optional
HasComponent	Variable	MinutesItemLife	Double	MTToolLife-Type	Optional
HasComponent	Variable	CutterStatus	UInteger	MTCutter-StatusType	Optional
HasComponent	Variable	PartCountItemLife	Double	MTToolLife-Type	Optional
Organizes	Object	<Folder>	MTCuttingTool-MeasurementType[]	FolderType	Optional
HasComponent	Variable	WearItemLife	Double	MTToolLife-Type	Optional

7.2.2.1 Referenced Properties and Objects

- `Indices::String:` TODO: We may want to create a special type for the indices.

7.2.3 Defintion of `MTCuttingItemsFolderType`

Table 12: `MTCuttingItemsFolderType` Definition

Attribute	Value				
BrowseName	MTCuttingItemsFolderType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modeling-Rule**
Subtype of FolderType (See [UA Part 05] Documentation)					
HasProperty	Variable	Count	Int32	PropertyType	Mandatory
Organizes	Object	<Folder>	MTCuttingToolLifeCycle-Type	FolderType	Mandatory

7.2.4 Defintion of `MTCuttingItemsFolderType`

Table 13: `MTCuttingItemsFolderType` Definition

Attribute	Value				
BrowseName	MTCuttingItemsFolderType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
HasProperty	Variable	Count	Int32	PropertyType	Mandatory

7.2.5 Defintion of `MTCuttingToolArchetypeType`

The `MTCuttingToolArchetypeType` *Information Model* represents the information about a of cutting tool that is common to all cutting toos of that kind. It contains only the constraints on the measurements and the limits of use, but as it deos not represent a physical cutting tool, there can be not measured values.

MTConnect Standard will adopt the ISO 13399 structure when formulating the vocabulary for Cutting Tool geometries and structure to be represented in the CuttingToolArchetype. The nominal values provided in the `MTCuttingToolLifeCycleType` section are only concerned with two aspects of the Cutting Tool, the Cutting Tool and the Cutting Item. The Tool Item, Adaptive Item, and Assembly Item will only be covered in the `MT-CuttingToolDefinitionType` section of this document since this section contains the full ISO 13399 information about a Cutting Tool.

Table 14: `MTCuttingToolArchetypeType` Definition

Attribute	Value				
BrowseName	MTCuttingToolArchetypeType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTAssetType (See Assets Documentation)					
HasProperty	Variable	Manufacturers	String[]	PropertyType	Optional
HasProperty	Variable	SerialNumber	String	PropertyType	Mandatory
HasProperty	Variable	ToolId	String	PropertyType	Mandatory
HasComponent	Object	CuttingToolLifecycle	MTCuttingToolLifeCycleType		Optional
HasComponent	Object	CuttingToolDefinition	MTCuttingToolDefinitionType		Optional

7.2.6 Defintion of `MTCuttingToolConstraintType`

Table 15: `MTCuttingToolConstraintType` Definition

Attribute	Value				
BrowseName	MTCuttingToolConstraintType				
IsAbstract	False				
ValueRank	-1				
DataType	Double				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of DataItemType (See [UA Part 08] Documentation)					
HasSubtype	VariableType	MTCuttingToolMeasurementType		See section 7.2.7	
HasProperty	Variable	Maximum	Double	PropertyType	Optional
HasProperty	Variable	Minimum	Double	PropertyType	Optional
HasProperty	Variable	Nominal	Double	PropertyType	Optional

7.2.7 Defintion of `MTCuttingToolMeasurementType`

The `MTCuttingToolMeasurementsType` is used for all measurements where the **`BrowseName`** provides the semantic information regarding the measurement type along with the `Code` ***Property*** which refers to the ISO 13399 term.

The measurements **`BrowseName`** will be taken from the *Element* name of the measurement. The reset of the information will be included per the MTConnect standard.

Table 16: `MTCuttingToolMeasurementType` Definition

Attribute	Value				
BrowseName	MTCuttingToolMeasurementType				
IsAbstract	True				
ValueRank	-1				
DataType	Double				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTCuttingToolConstraintType (See section 7.2.6)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	EngineringUnits	EUInformation	PropertyType	Optional
HasProperty	Variable	NativeUnits	String	PropertyType	Optional
HasProperty	Variable	SignificantDigits	UInt32	PropertyType	Optional
HasProperty	Variable	Units	String	PropertyType	Optional
HasMTClassType	Object	<MTDataItemClass>	MTDataItemClassType		Mandatory

7.2.7.1 Referenced Properties and Objects

- `Code::String:` Refers to the ISO 13399 constraints (TODO: Need reference to the ISO specification for the codes).

7.2.8 Defintion of `MTCuttingToolDefinitionType`

The CuttingToolDefinition contains the detailed structure of the Cutting Tool. The information contained in this element will be static during its lifecycle. Currently we are referring to the external ISO 13399 standard to provide the complete definition and composition of the Cutting Tool.

Table 17: `MTCuttingToolDefinitionType` Definition

Attribute	**Value**				
BrowseName	MTCuttingToolDefinitionType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modeling-Rule**
Subtype of BaseObjectType (See [UA Part 05] Documentation)					
HasProperty	Variable	Format	CuttingToolDefintion-FormatDataType	PropertyType	Optional
HasComponent	Object	Data	FileType		Mandatory

7.2.8.1 Referenced Properties and Objects

- **Allowable Values** for `CuttingToolDefintionFormatDataType`

Table 18: `CuttingToolDefintionFormatDataType` Enumeration

Name	**Index**
XML	0
EXPRESS	1
TEXT	2
UNDEFINED	3

7.2.9 Defintion of `MTCuttingToolLifeCycleType`

The life cycle refers to the data pertaining to the application or the use of the tool. This data is provided by various pieces of equipment (i.e. machine tool, presetter) and statistical process control applications. Life cycle data will not remain static, but will change periodically when a tool is used or measured. The life cycle has three conceptual parts; tool and Cutting Item identity, properties, and measurements.

The CuttingToolLifeCycle contains data for the entire tool assembly. The specific Cutting Items that are part of the CuttingToolLifeCycle are contained in the CuttingItems element. Each `MTCuttingItem` has similar properties as the assembly; identity, properties, and measurements.

Table 19: `MTCuttingToolLifeCycleType` Definition

Attribute	Value				
BrowseName	MTCuttingToolLifeCycleType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**Type-Definition**	**Modeling-Rule**
Subtype of BaseObjectType (See [UA Part 05] Documentation)					
HasProperty	Variable	ConnectionCode-MachineSide	String	PropertyType	Optional
HasProperty	Variable	ProgramToolGroup	String	PropertyType	Optional
HasProperty	Variable	ProgramToolNumber	Int32	PropertyType	Optional
HasComponent	Variable	CutterStatus	UInteger	MTCutter-StatusType	Mandatory
HasComponent	Variable	ToolLifePartCount	Double	MTToolLife-Type	Optional
HasComponent	Variable	ToolLifeWear	Double	MTToolLife-Type	Optional
HasComponent	Variable	ProcessFeedRate	Double	MTCutting-Tool-Constraint-Type	Optional
HasComponent	Variable	ReconditionCount	Int32	MT-Recondition-CountType	Optional
Organizes	Object	<Folder>	MTCuttingItem-Type[]	FolderType	Optional
HasComponent	Variable	ProcessSpindleSpeed	Double	MTCutting-Tool-Constraint-Type	Optional
Organizes	Object	<Folder>	MTCuttingTool-MeasurementType[]	FolderType	Optional
HasComponent	Variable	Location	Int32	MTLocation-Type	Optional
HasComponent	Variable	ToolLifeMinutes	Double	MTToolLife-Type	Optional

7.2.10 Defintion of **MTCuttingToolType**

Table 20: `MTCuttingToolType` Definition

Attribute	Value				
BrowseName	MTCuttingToolType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTAssetType (See Assets Documentation)					
HasProperty	Variable	Manufacturers	String[]	PropertyType	Optional
HasProperty	Variable	SerialNumber	String	PropertyType	Mandatory
HasProperty	Variable	ToolId	String	PropertyType	Mandatory
HasComponent	Object	CuttingToolArchitype	MTCuttingToolArchetypeType		Optional
HasComponent	Object	CuttingToolLifecycle	MTCuttingToolLifeCycleType		Mandatory

7.2.11 Defintion of **MTLocationType**

Table 21: `MTLocationType` Definition

Attribute	Value				
BrowseName	MTLocationType				
IsAbstract	False				
ValueRank	-1				
DataType	Int32				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of DataItemType (See [UA Part 08] Documentation)					
HasProperty	Variable	NegativeOverlap	Int32	PropertyType	Optional
HasProperty	Variable	PositiveOverlap	Int32	PropertyType	Optional
HasProperty	Variable	Type	MTLocationDataType	PropertyType	Mandatory

7.2.11.1 Referenced Properties and Objects

- **Allowable Values** for `MTLocationDataType`

Table 22: `MTLocationDataType` Enumeration

Name	Index
CRIB	0
POT	1
STATION	2

645 7.2.12 Defintion of `MTReconditionCountType`

Table 23: `MTReconditionCountType` Definition

Attribute	Value				
BrowseName	MTReconditionCountType				
IsAbstract	False				
ValueRank	-1				
DataType	Int32				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of DataItemType (See [UA Part 08] Documentation)					
HasProperty	Variable	MaximumCount	Int32	PropertyType	Optional

646 7.2.13 Defintion of `MTToolLifeType`

Table 24: `MTToolLifeType` Definition

Attribute	Value				
BrowseName	MTToolLifeType				
IsAbstract	False				
ValueRank	-1				
DataType	Double				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of DataItemType (See [UA Part 08] Documentation)					
HasProperty	Variable	CountDirection	CountDirectionDataType	PropertyType	Mandatory
HasProperty	Variable	Initial	Double	PropertyType	Optional
HasProperty	Variable	Limit	Double	PropertyType	Optional
HasProperty	Variable	MTType	String	PropertyType	Mandatory
HasProperty	Variable	Warning	Double	PropertyType	Optional

647 7.2.13.1 Referenced Properties and Objects

648 • **Allowable Values** for `CountDirectionDataType`

Table 25: `CountDirectionDataType` Enumeration

Name	Index
DOWN	0
UP	1

649 7.3 Measurements

650 7.3.1 Defintion of `CommonMeasurementType`

Table 26: `CommonMeasurementType` Definition

Attribute	**Value**				
BrowseName	CommonMeasurementType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**

651 7.3.2 Defintion of `CuttingToolClassType`

Table 27: `CuttingToolClassType` Definition

Attribute	**Value**				
BrowseName	CuttingToolClassType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemClassType (See Data Item Types Documentation)					
HasSubtype	ObjectType	BodyDiameterMaxClassType		See section 7.3.3	
HasSubtype	ObjectType	BodyLengthMaxClassType		See section 7.3.4	
HasSubtype	ObjectType	CuttingDiameterMaxType		See section 7.3.5	
HasSubtype	ObjectType	CuttingItemClassType		See section 7.3.6	
HasSubtype	ObjectType	DepthOfCutMaxClassType		See section 7.3.27	
HasSubtype	ObjectType	FlangeDiameterMaxClassType		See section 7.3.28	
HasSubtype	ObjectType	FunctionalLengthClassType		See section 7.3.29	
HasSubtype	ObjectType	OverallToolLengthClassType		See section 7.3.30	
HasSubtype	ObjectType	ProtrudingLengthClassType		See section 7.3.31	
HasSubtype	ObjectType	ShankDiameterClassType		See section 7.3.32	
HasSubtype	ObjectType	ShankHeightClassType		See section 7.3.33	
HasSubtype	ObjectType	ShankLengthClassType		See section 7.3.34	
HasSubtype	ObjectType	UsableLengthMaxClassType		See section 7.3.35	
HasSubtype	ObjectType	WeightClassType		See section 7.3.36	
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

652 7.3.3 Defintion of `BodyDiameterMaxClassType`

Table 28: `BodyDiameterMaxClassType` Definition

Attribute	Value				
BrowseName	BodyDiameterMaxClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingToolClassType (See section 7.3.2)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

653 7.3.4 Defintion of `BodyLengthMaxClassType`

Table 29: `BodyLengthMaxClassType` Definition

Attribute	Value				
BrowseName	BodyLengthMaxClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingToolClassType (See section 7.3.2)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

654 7.3.5 Defintion of `CuttingDiameterMaxType`

Table 30: `CuttingDiameterMaxType` Definition

Attribute	Value				
BrowseName	CuttingDiameterMaxType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingToolClassType (See section 7.3.2)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

655 7.3.6 Defintion of `CuttingItemClassType`

Table 31: `CuttingItemClassType` Definition

Attribute	Value				
BrowseName	CuttingItemClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingToolClassType (See section 7.3.2)					
HasSubtype	ObjectType	ChamferFlatLengthClassType		See section 7.3.7	
HasSubtype	ObjectType	ChamferWidthClassType		See section 7.3.8	
HasSubtype	ObjectType	CornerRadiusClassType		See section 7.3.9	
HasSubtype	ObjectType	CuttingDiameterClassType		See section 7.3.10	
HasSubtype	ObjectType	CuttingEdgeLengthClassType		See section 7.3.11	
HasSubtype	ObjectType	CuttingHeightClassType		See section 7.3.12	
HasSubtype	ObjectType	CuttingItemFunctionalLengthClassType		See section 7.3.13	
HasSubtype	ObjectType	CuttingItemWeightClassType		See section 7.3.14	
HasSubtype	ObjectType	DriveAngleClassType		See section 7.3.15	
HasSubtype	ObjectType	FlangeDiameterClassType		See section 7.3.16	
HasSubtype	ObjectType	FunctionalWidthClassType		See section 7.3.17	
HasSubtype	ObjectType	IncribedCircleDiameterClassType		See section 7.3.18	
HasSubtype	ObjectType	InsertWidthClassType		See section 7.3.19	
HasSubtype	ObjectType	PointAngleClassType		See section 7.3.20	
HasSubtype	ObjectType	StepDiameterLengthClassType		See section 7.3.21	
HasSubtype	ObjectType	StepIncludedAngleClassType		See section 7.3.22	
HasSubtype	ObjectType	ToolCuttingEdgeAngleClassType		See section 7.3.23	
HasSubtype	ObjectType	ToolLeadAngleClassType		See section 7.3.24	
HasSubtype	ObjectType	ToolOrientationClassType		See section 7.3.25	
HasSubtype	ObjectType	WiperEdgeLengthClassType		See section 7.3.26	

656 7.3.7 Defintion of `ChamferFlatLengthClassType`

Table 32: `ChamferFlatLengthClassType` Definition

Attribute	Value				
BrowseName	ChamferFlatLengthClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingItemClassType (See section 7.3.6)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

657 7.3.8 Defintion of `ChamferWidthClassType`

Table 33: `ChamferWidthClassType` Definition

Attribute	Value				
BrowseName	ChamferWidthClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingItemClassType (See section 7.3.6)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

658 7.3.9 Defintion of `CornerRadiusClassType`

Table 34: `CornerRadiusClassType` Definition

Attribute	Value				
BrowseName	CornerRadiusClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingItemClassType (See section 7.3.6)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

659 7.3.10 Defintion of `CuttingDiameterClassType`

Table 35: `CuttingDiameterClassType` Definition

Attribute	Value				
BrowseName	CuttingDiameterClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingItemClassType (See section 7.3.6)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

660 7.3.11 Defintion of `CuttingEdgeLengthClassType`

Table 36: `CuttingEdgeLengthClassType` Definition

Attribute	Value				
BrowseName	CuttingEdgeLengthClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingItemClassType (See section 7.3.6)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

661 7.3.12 Defintion of `CuttingHeightClassType`

Table 37: `CuttingHeightClassType` Definition

Attribute	Value				
BrowseName	CuttingHeightClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingItemClassType (See section 7.3.6)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

662 7.3.13 Defintion of `CuttingItemFunctionalLengthClassType`

Table 38: `CuttingItemFunctionalLengthClassType` Definition

Attribute	Value				
BrowseName	CuttingItemFunctionalLengthClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingItemClassType (See section 7.3.6)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

663 7.3.14 Defintion of `CuttingItemWeightClassType`

Table 39: `CuttingItemWeightClassType` Definition

Attribute	Value				
BrowseName	CuttingItemWeightClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingItemClassType (See section 7.3.6)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

664 7.3.15 Defintion of `DriveAngleClassType`

Table 40: `DriveAngleClassType` Definition

Attribute	Value				
BrowseName	DriveAngleClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingItemClassType (See section 7.3.6)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

665 7.3.16 Defintion of `FlangeDiameterClassType`

Table 41: `FlangeDiameterClassType` Definition

Attribute	Value				
BrowseName	FlangeDiameterClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingItemClassType (See section 7.3.6)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

7.3.17 Defintion of `FunctionalWidthClassType`

Table 42: `FunctionalWidthClassType` Definition

Attribute	Value				
BrowseName	FunctionalWidthClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingItemClassType (See section 7.3.6)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

7.3.18 Defintion of `IncribedCircleDiameterClassType`

Table 43: `IncribedCircleDiameterClassType` Definition

Attribute	Value				
BrowseName	IncribedCircleDiameterClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingItemClassType (See section 7.3.6)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

7.3.19 Defintion of `InsertWidthClassType`

Table 44: `InsertWidthClassType` Definition

Attribute	Value				
BrowseName	InsertWidthClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingItemClassType (See section 7.3.6)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

669 7.3.20 Defintion of `PointAngleClassType`

Table 45: `PointAngleClassType` Definition

Attribute	Value				
BrowseName	PointAngleClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingItemClassType (See section 7.3.6)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

670 7.3.21 Defintion of `StepDiameterLengthClassType`

Table 46: `StepDiameterLengthClassType` Definition

Attribute	Value				
BrowseName	StepDiameterLengthClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingItemClassType (See section 7.3.6)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

671 7.3.22 Defintion of `StepIncludedAngleClassType`

Table 47: `StepIncludedAngleClassType` Definition

Attribute	Value				
BrowseName	StepIncludedAngleClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingItemClassType (See section 7.3.6)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

672 7.3.23 Defintion of `ToolCuttingEdgeAngleClassType`

Table 48: `ToolCuttingEdgeAngleClassType` Definition

Attribute	Value				
BrowseName	ToolCuttingEdgeAngleClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingItemClassType (See section 7.3.6)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

673 7.3.24 Defintion of `ToolLeadAngleClassType`

Table 49: `ToolLeadAngleClassType` Definition

Attribute	Value				
BrowseName	ToolLeadAngleClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingItemClassType (See section 7.3.6)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

674 7.3.25 Defintion of `ToolOrientationClassType`

Table 50: `ToolOrientationClassType` Definition

Attribute	Value				
BrowseName	ToolOrientationClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingItemClassType (See section 7.3.6)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

675 7.3.26 Defintion of `WiperEdgeLengthClassType`

Table 51: `WiperEdgeLengthClassType` Definition

Attribute	Value				
BrowseName	WiperEdgeLengthClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingItemClassType (See section 7.3.6)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

676 7.3.27 Defintion of `DepthOfCutMaxClassType`

Table 52: `DepthOfCutMaxClassType` Definition

Attribute	Value				
BrowseName	DepthOfCutMaxClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingToolClassType (See section 7.3.2)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

677 7.3.28 Defintion of `FlangeDiameterMaxClassType`

Table 53: FlangeDiameterMaxClassType Definition

Attribute	Value				
BrowseName	FlangeDiameterMaxClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingToolClassType (See section 7.3.2)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

678 ### 7.3.29 Defintion of `FunctionalLengthClassType`

Table 54: FunctionalLengthClassType Definition

Attribute	Value				
BrowseName	FunctionalLengthClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingToolClassType (See section 7.3.2)					
HasProperty	Variable	FunctionalLength	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

679 ### 7.3.30 Defintion of `OverallToolLengthClassType`

Table 55: OverallToolLengthClassType Definition

Attribute	Value				
BrowseName	OverallToolLengthClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingToolClassType (See section 7.3.2)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

680 ### 7.3.31 Defintion of `ProtrudingLengthClassType`

Table 56: `ProtrudingLengthClassType` Definition

Attribute	Value				
BrowseName	ProtrudingLengthClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingToolClassType (See section 7.3.2)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

681 7.3.32 Defintion of `ShankDiameterClassType`

Table 57: `ShankDiameterClassType` Definition

Attribute	Value				
BrowseName	ShankDiameterClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingToolClassType (See section 7.3.2)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

682 7.3.33 Defintion of `ShankHeightClassType`

Table 58: `ShankHeightClassType` Definition

Attribute	Value				
BrowseName	ShankHeightClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingToolClassType (See section 7.3.2)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

683 7.3.34 Defintion of `ShankLengthClassType`

Table 59: `ShankLengthClassType` Definition

Attribute	**Value**				
BrowseName	ShankLengthClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingToolClassType (See section 7.3.2)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

684 7.3.35 Defintion of `UsableLengthMaxClassType`

Table 60: `UsableLengthMaxClassType` Definition

Attribute	**Value**				
BrowseName	UsableLengthMaxClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingToolClassType (See section 7.3.2)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

685 7.3.36 Defintion of `WeightClassType`

Table 61: `WeightClassType` Definition

Attribute	**Value**				
BrowseName	WeightClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of CuttingToolClassType (See section 7.3.2)					
HasProperty	Variable	Code	String	PropertyType	Mandatory
HasProperty	Variable	Units	String	PropertyType	Mandatory

686 7.4 Assets Profile

687 7.4.1 Defintion of «`HasCuttingToolArchetype`»

8 Profiles and Namespaces

8.1 Namespace Metadata

Table 62 defines the ***namespace metadata*** for this specification. The ***Object*** is used to provide 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 [UA Part 05] 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 [UA Part 05].

The version information is also provided as part of the **`ModelTableEntry`** in the ***UANodeSet*** XML file. The ***UANodeSet*** XML schema is defined in [UA Part 06].

Table 62: `NamespaceMetadata` *Object* for this Specification

Attribute	**Value**		
BrowseName	http://www.opcfoundation.org/UA/MTConnect/2.0/		
References	**BrowseName**	**DataType**	**Value**
HasProperty	NamespaceUri	String	http://www.opcfoundation.org/UA/MTConnect/2.0/
HasProperty	NamespaceVersion	String	2.0
HasProperty	NamespacePublicationDate	DateTime	2018-10-31T00:00:00
HasProperty	IsNamespaceSubset	Boolean	False
HasProperty	StaticNodeIdTypes	IdType[]	[0]
HasProperty	StaticNumericNodeIdRange	NumericRange[]	[1:1073741824]
HasProperty	StaticStringNodeIdPattern	String	

8.2 Conformance Units and Profiles

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

8.2.1 Server

Table 63 defines the ***Server*** based ***ConformanceUnits***.

Table 63: MTConnect ***Server*** Information Model

Conformance Unit	Description	Optional/ Mandatory
MTConnect Base Functionality	The server supports the ***BaseObjectModel***. This includes exposing all mandatory objects, variables, methods, and data types.	M
Availability	The Server must support the `Availability` *DataItem* to indicate if data is available from the device.	M
Device	The Server has at least one root `MTDeviceType`	M
AssetChanged Data Item	The Server must support the MTConnect AssetChanged and AssetRemoved data items	O
Message	The Server must support the MTConnect Message data item and publish `MTMessageEventType` **`Event`**s	M
Condition	The server must support the MTConnect `MTConditionType` type and provide correct activation states	M
Condition Branches	The server must support MTConnect `MTConditionType` condition branches to represent multiple MTConnect Condition parallel activations	O
Three Space Sample	The server must support the `MTThreeSpaceSampleType` data type to provide a spacial coordinate	M
MTHasClassType and MTHasSubClassType	The server must have `MTSampleType`, `MTStringEventType` `MTMessageType`, `MTNumericEventType`, and `MTControlledVariableType` with relationships to the MTConnect Class types associated with the MTConnect `DataItem` `type` and `subType`	M
MTConnect meta data	DataItems represented in OPC UA must have the full meta data required by the MTConnect standard for all attributes	M
Engineer Units	All `MTSampleType` data items must have the **`EngineeringUnits`** follow the prescribed Units as specified in the MTConnect standard.	M

8.2.2 Client

Table 64 defines the ***Client*** based ***ConformanceUnits***.

Table 64: MTConnect ***Client*** Information Model

Conformance Unit	Description	Optional/ Mandatory
MTConnect Base Functionality	The client supports the ***BaseObjectModel***. This includes exposing all mandatory objects, variables, methods, and data types.	M
Availability	The client must interpret the `Availability` *DataItem* to indicate if data is available from the device.	M

8.3 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 glsBrowseName shall have the ***namespace*** of the standards body although the ***namespace*** of the **NodeId** reflects something else, for example the **EngineeringUnits** ***Property***. All **NodeIds** of ***Nodes*** not defined in this specification shall not use the standard ***namespaces***.

Table 65 provides a list of mandatory and optional namespaces used in an MTConnect OPC UA ***Server***.

Table 65: Namespaces used in a MTConnect Server

NamespaceURI	Description	Use
http://www.opcfoundation.org/UA/	***Namespace*** for **NodeIds** and **BrowseNames** defined in the OPC UA specification. This ***namespace*** shall have ***namespace*** index 0.	Mandatory
Local Server URI	***Namespace*** for nodes defined in the local server. This may include types and instances used in an ***AutoID*** Device represented by the ***Server***. This ***namespace*** shall have ***namespace*** index 1.	Mandatory
http://www.opcfoundation.org/UA/MTConnect/2.0/	***Namespace*** for **NodeIds** and **BrowseNames** defined in this specification. The ***namespace*** index is ***Server*** specific.	Mandatory
Vendor specific types	A ***Server*** may provide vendor-specific types like types derived from ***ObjectTypes*** defined in this specification in a vendor-specific ***namespace***.	Optional
Vendor specific instances	A ***Server*** provides vendor-specific instances of the standard types or vendor-specific instances of vendor-specific types in a vendor-specific ***namespace***. It is recommended to separate vendor specific types and vendor specific instances into two or more ***namespaces***.	Mandatory

Table 66 provides a list of ***namespaces*** and their index used for **BrowseNames** in this specification. The default ***namespace*** of this specification is not listed since all **BrowseNames** without prefix use this default ***namespace***.

Table 66: Namespaces used used in this specification

NamespaceURI	Namespace Index	Example
http://www.opcfoundation.org/UA/	0	0:EngineeringUnits
http://www.opcfoundation.org/UA/MTConnect/2.0/	1	1:MTDevice

Annex A MTConnect Namespace and Mappings (normative)

A.1 Namespace and identifiers for MTConnect Information Model

This appendix defines the numeric identifiers for all of the numeric NodeIds defined in this specification. The identifiers are specified in a CSV file with the following syntax:

```
<SymbolName>, <Identifier>, <NodeClass>
```

Where the ***SymbolName*** is either the **BrowseName** of a Type ***Node*** or the ***BrowsePath*** for an ***Instance Node*** that appears in the specification and the Identifier is the numeric value for the **NodeId**.

The ***BrowsePath*** for an Instance ***Node*** is constructed by appending the **BrowseName** of the instance ***Node*** to the **BrowseName** for the containing instance or type. An underscore character is used to separate each **BrowseName** in the path. Let's take for example, the MTComponentType **ObjectType** Node which has the NativeName ***Property***. The **Name** for the NativeName ***InstanceDeclaration*** within the MTComponentType declaration is as follows: MTComponentType_NativeName.

The CSV associated with this version of the standard can be found here:

http://www.opcfoundation.org/UA/schemas/MTConnect/2.0/MTConnect.NodeIds.csv

NOTE The latest CSV that is compatible with this version of the standard can be found here:

http://www.opcfoundation.org/UA/schemas/MTConnect/MTConnect.NodeIds.csv

A computer processible version of the complete ***Information Model*** defined in this specification is also provided. It follows the XML ***Information Model*** schema syntax defined in OPC [UA Part 06].

The information schema for this version of the standard, including all errata, can be found at the following URL:

http://www.opcfoundation.org/UA/schemas/MTConnect/2.0/Opc.Ua.MTConnect.NodeSet2.xml

NOTE: The latest information schema for this version of the standard, including all errata, can be found at the following URL:

http://www.opcfoundation.org/UA/schemas/MTConnect/Opc.Ua.MTConnect.NodeSet2.xml