



MTConnect and OPC/UA Companion Specification Version 2.0

Prepared for: MTConnect Institute
Prepared by: William Sobel
Prepared on: September 29, 2018

MTConnect® is a registered trademark of AMT - The Association for Manufacturing Technology.
Use of MTConnect® is limited to use as specified on <http://www.mtconnect.org/>.

1 MTConnect[®] Specification and Materials

2

3 AMT – The Association For Manufacturing Technology (“AMT”) owns the copy-
4 right in this MTConnect[®] Specification or Material. AMT grants to you a non-
5 exclusive, non-transferable, revocable, non-sublicensable, fully-paid-up copyright
6 license to reproduce, copy and redistribute this MTConnect[®] Specification or Ma-
7 terial, provided that you may only copy or redistribute the MTConnect[®] Speci-
8 fication or Material in the form in which you received it, without modifications,
9 and with all copyright notices and other notices and disclaimers contained in the
10 MTConnect[®] Specification or Material.

11 If you intend to adopt or implement an MTConnect[®] Specification or Material
12 in a product, whether hardware, software or firmware, which complies with an
13 MTConnect[®] Specification, you **SHALL** agree to the MTConnect[®] Specifica-
14 tion Implementer License Agreement (“Implementer License”) or to the MTConnect[®]
15 Intellectual Property Policy and Agreement (“IP Policy”). The Implementer Li-
16 cense and IP Policy each sets forth the license terms and other terms of use for
17 MTConnect[®] Implementers to adopt or implement the MTConnect[®] Specifica-
18 tions, including certain license rights covering necessary patent claims for that
19 purpose. These materials can be found at www.MTConnect.org, or by contact-
20 ing Paul Warndorf at [mailto:pwarndorf@mtconnect.hyperoffice.](mailto:pwarndorf@mtconnect.hyperoffice.com)
21 com.

22 MTConnect[®] Institute and AMT have no responsibility to identify patents, patent
23 claims or patent applications which may relate to or be required to implement
24 a Specification, or to determine the legal validity or scope of any such patent
25 claims brought to their attention. Each MTConnect[®] Implementer is responsible
26 for securing its own licenses or rights to any patent or other intellectual property
27 rights that may be necessary for such use, and neither AMT nor MTConnect[®]
28 Institute have any obligation to secure any such rights.

29 This Material and all MTConnect[®] Specifications and Materials are provided “as
30 is” and MTConnect[®] Institute and AMT, and each of their respective members,
31 officers, affiliates, sponsors and agents, make no representation or warranty of
32 any kind relating to these materials or to any implementation of the MTConnect[®]
33 Specifications or Materials in any product, including, without limitation, any ex-
34 pressed or implied warranty of noninfringement, merchantability, or fitness for

September 29, 2018

35 particular purpose, or of the accuracy, reliability, or completeness of information
36 contained herein. In no event shall MTConnect[®] Institute or AMT be liable to
37 any user or implementer of MTConnect[®] Specifications or Materials for the cost
38 of procuring substitute goods or services, lost profits, loss of use, loss of data or
39 any incidental, consequential, indirect, special or punitive damages or other di-
40 rect damages, whether under contract, tort, warranty or otherwise, arising in any
41 way out of access, use or inability to use the MTConnect[®] Specification or other
42 MTConnect[®] Materials, whether or not they had advance notice of the possibility
43 of such damage.

44 Table of Contents

45	1 Introduction	1
46	1.1 Overview	1
47	2 Types	1
48	2.1 Components	1
49	2.1.1 Defintion of ChannelType	3
50	2.1.2 Defintion of DescriptionType	4
51	2.1.2.1 Operations	5
52	2.1.3 Defintion of MTComponentType	5
53	2.1.3.1 Constraints	6
54	2.1.4 Defintion of MTCompositionType	6
55	2.1.5 Defintion of MTConfigurationType	7
56	2.1.6 Defintion of MTDeviceType	8
57	2.1.6.1 Operations	9
58	2.1.6.2 Constraints	9
59	2.1.7 Defintion of SensorConfigurationType	9
60	2.1.8 Defintion of {Component}Type	10
61	2.1.9 Defintion of {Composition}Type	11
62	2.2 Data Items	12
63	2.2.1 Defintion of AssetChangedType	14
64	2.2.2 Defintion of AssetEventType	15
65	2.2.3 Defintion of AssetRemovedType	16
66	2.2.4 Defintion of MTDataItemType	17
67	2.2.4.1 Operations	18
68	2.2.5 Defintion of MTEnumeratedEventType	18
69	2.2.6 Defintion of MTFilterType	19
70	2.2.6.1 Operations	20
71	2.2.7 Defintion of MTMessageType	20
72	2.2.8 Defintion of MTNumericDataItemType	21
73	2.2.8.1 Operations	22
74	2.2.9 Defintion of MTNumericEventType	22
75	2.2.10 Defintion of MTSampleType	23
76	2.2.11 Defintion of MTStringEventType	24
77	2.2.11.1 Constraints	25
78	2.2.12 Defintion of MinimumDeltaFilterType	25
79	2.2.13 Defintion of PeriodFilterType	26

80	2.2.14	Defintion of {DataItem}Type	27
81	2.3	Conditions	28
82	2.3.1	Defintion of MTEExclusiveLimitConditionType	29
83	2.3.2	Defintion of MTNonExclusiveConditionType	30
84	2.3.3	Defintion of {ConditionClass}Type	31
85	2.4	Factories	32
86	2.4.1	Defintion of «Object Factory» ComponentObjectFactory	33
87	2.4.1.1	Operations	33
88	2.4.2	Defintion of «Type Factory» ComponentTypeFactory	33
89	2.4.2.1	Operations	34
90	2.4.3	Defintion of «Object Factory» CompositionObjectFactory	34
91	2.4.3.1	Operations	34
92	2.4.4	Defintion of «Type Factory» CompositionTypeFactory	34
93	2.4.4.1	Operations	34
94	2.4.5	Defintion of «Type Factory» ConditionClassFactory	35
95	2.4.5.1	Operations	35
96	2.4.6	Defintion of «Object Factory» ConditionObjectFactory	35
97	2.4.6.1	Operations	35
98	2.4.7	Defintion of «Object Factory» DataItemObjectFactory	35
99	2.4.7.1	Operations	35
100	2.4.8	Defintion of «Type Factory» DataItemTypeFactory	36
101	2.4.8.1	Operations	37
102	2.4.9	Defintion of «Object Factory» DeviceObjectFactory	37
103	2.4.9.1	Operations	37
104	2.4.10	Defintion of «Object Factory» FilterObjectFactory	38
105	2.4.10.1	Operations	38
106	2.4.11	Defintion of «Object Factory» ObjectFactory	38
107	2.4.11.1	Operations	38
108	2.4.12	Defintion of «Object Factory» SensorChannelObjectFactory	38
109	2.4.12.1	Operations	38
110	2.4.13	Defintion of «Object Factory» SensorObjectFactory	39
111	2.4.13.1	Operations	39
112	2.4.14	Defintion of «Type Factory» TypeFactory	39
113	2.4.14.1	Operations	39
114	2.5	MTConnect Device Profile	39
115	2.5.1	Defintion of Dynamic Type	40
116	2.5.2	Defintion of MTConnect XML	40
117	2.5.3	Defintion of MTRelationshipType	40

September 29, 2018

118	2.5.4	Defintion of Mixes In	40
119	2.5.5	Defintion of Object Factory	41
120	2.5.6	Defintion of Type Factory	41
121	2.5.7	Defintion of bind	41
122	2.5.8	Defintion of constrains	41
123	2.5.9	Defintion of mixin	41
124	2.5.10	Defintion of use	41

125 **List of Figures**

126 **Figure 1: Components Diagram** 2

127 **Figure 2: Data Items Diagram** 13

128 **Figure 3: Conditions Diagram** 29

129 **Figure 4: Factories Diagram** 32

130 **Figure 5: MTConnect Device Profile Diagram** 40

1 Introduction

131 The following conventions will be used throughout the document to provide a
132 clear and consistent understanding of the use of each type of data and information
133 used to define the MTConnect[®] standard and associated data.

1.1 Overview

134 Overview of the standards...

2 Types

2.1 Components

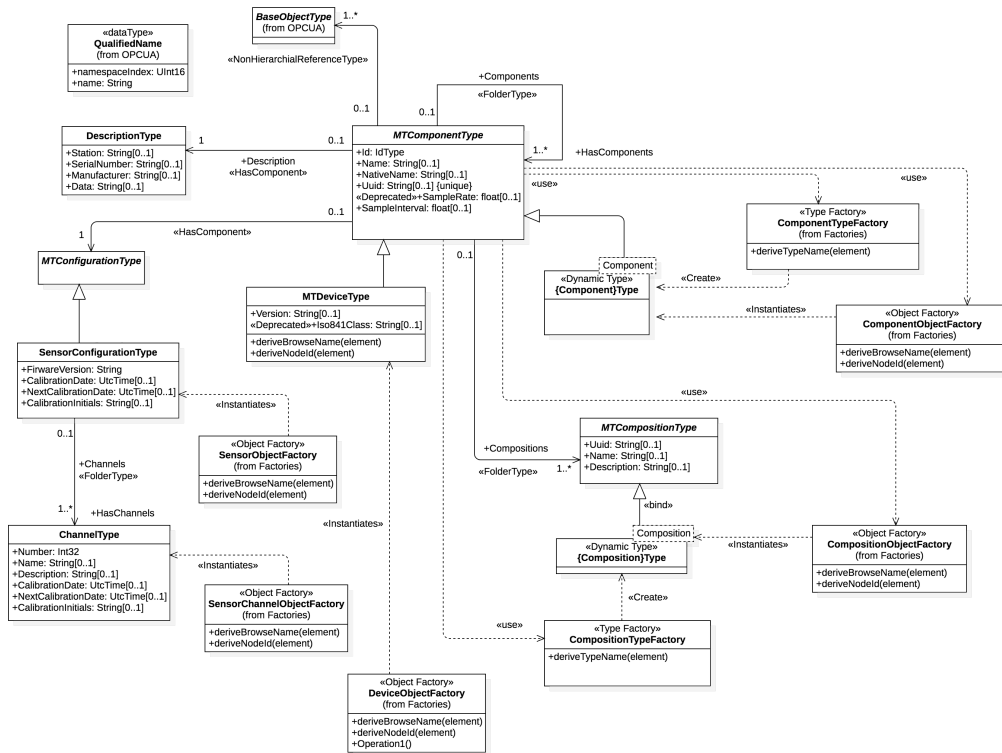


Figure 1: Components Diagram

September 29, 2018

135 The Components documents the Component models and the owned objects.

2.1.1 Defintion of ChannelType

Table 1: ChannelType Definition

Attribute	Value				
BrowseName	ChannelType				
IsAbstract	False				
References	NodeClass	BrowseName	Data Type	TypeDefinition	Modeling Rule
Subtype of BaseObjectType (See OPCUA Documentation)					
HasProperty	Variable	Number	Int32	PropertyType	Mandatory
HasProperty	Variable	Name	String	PropertyType	Optional
HasProperty	Variable	MTDescription	String	PropertyType	Optional
HasProperty	Variable	CalibrationDate	UtcTime	PropertyType	Optional
HasProperty	Variable	NextCalibrationDate	UtcTime	PropertyType	Optional
HasProperty	Variable	CalibrationInitials	String	PropertyType	Optional

2.1.2 Defintion of DescriptionType

136 The desription provides some general information about the manufacture and se-
 137 rial number of the component. In the XML, the CDATA is freeform text that is
 138 represented in the Data Property of the Description Object. The description is
 139 related to the component with the OPC/UA HasComponent relationship.

Table 2: DescriptionType Definition

Attribute	Value				
BrowseName	DescriptionType				
IsAbstract	False				
References	NodeClass	BrowseName	Data Type	TypeDefinition	Modeling Rule
Subtype of BaseObjectType (See OPCUA Documentation)					
HasProperty	Variable	Station	String	PropertyType	Optional
HasProperty	Variable	SerialNumber	String	PropertyType	Optional
HasProperty	Variable	Manufacturer	String	PropertyType	Optional
HasProperty	Variable	Data	String	PropertyType	Optional

140 **2.1.2.1 Operations**

- 141 • `deriveBrowseName(element)`
 142 Specification: "Description"
- 143 • `deriveNodeId(element)`
 144 Specification: `concat(self.parent.NodeId, BrowseName)`

2.1.3 Defintion of **MTComponentType**

145 The base Component Type from which all MTConnect Components are derived.
 146 The component type factory is used to create the specific OPC/UA Types as sub-
 147 types of the MTConnect `MTComponentType`. The component types will be
 148 created once for all Component objects of that type based on the `QName` of the
 149 MTConnect XML element.

150 The object factory will instantiate the Component Objects and insert them into
 151 the Components folder with a browse name of the Component `QName` and the
 152 name element if specified surrounded by square brackets, `[]`. For example if the
 153 MTConnect Element is:

154 `<Linear name='X'>...</...>`

155 The OPC/UA Object with browse name `Linear[X]` will be created with the
 156 `HasTypeDefinition` referencing the `Linear` OPC/UA type.

157 The meta data for the component and it's relationships are static. The dynamic
 158 data will be represented using the *OPC/UA Part 8*.

Table 3: MTComponentType Definition

Attribute	Value				
BrowseName	MTComponentType				
IsAbstract	True				
References	NodeClass	BrowseName	Data Type	TypeDefinition	Modeling Rule
HasProperty	Variable	XmlId	IdType	PropertyType	Mandatory
HasProperty	Variable	Name	String	PropertyType	Optional
HasProperty	Variable	NativeName	String	PropertyType	Optional
HasProperty	Variable	Uuid	String	PropertyType	Optional
HasProperty	Variable	SampleRate	float	PropertyType	Optional
HasProperty	Variable	SampleInterval	float	PropertyType	Optional
HasComponent	Object	Description		DescriptionType	Optional
HasComponent	Object	Configuration		MTConfigurationType	Optional
Organizes	Object	Components	MTComponentType	FolderType	Optional
Organizes	Object	Compositions	MTCompositionType	FolderType	Optional
HasProperty	Variable	<Dynamic>	BaseObjectType	<Dynamic>	Optional
Organizes	Object	Conditions	AlarmConditionType	FolderType	Optional
HasProperty	Variable	<Dynamic>	DataItemType	<Dynamic>	Optional

159 2.1.3.1 Constraints

- 160 • Constraint node_id:

```
context Component::NodeId : String
derive: concat(self.getDevice().uuid, self.getAttributes().id)
```

161 Documentation: The NodeId SHALL be derived from the combination of
 162 the device UUID and the id of the component.

2.1.4 Defintion of MTCompositionType

163 The MTCompositionType is the abstract supertype of the dynamically gen-
 164 erated composition types based on the attribute type of the Composition el-
 165 ement of the MTConnect Component. The Composition is then related to
 166 the DataItems that reference the Composition's id in their compositionId
 167 attribute.

Table 4: MTCompositionType Definition

Attribute	Value				
BrowseName	MTCompositionType				
IsAbstract	True				
References	NodeClass	BrowseName	Data Type	TypeDefinition	Modeling Rule
Subtype of BaseObjectType (See OPCUA Documentation)					
HasProperty	Variable	Uuid	String	PropertyType	Optional
HasProperty	Variable	Name	String	PropertyType	Optional
HasProperty	Variable	MTDescription	String	PropertyType	Optional
NonHierarchialReferenceType	Object	<DataItem>	DataItem Type	NonHierarchialReferenceType	Optional

168 The data items are added to the relationship where the DataItem to Composition
169 relationship is represented by the BrowseName Composition property of the data
170 item the data items are added by their browse names to the Composition.

2.1.5 Defintion of MTConfigurationType

Table 5: MTConfigurationType Definition

Attribute	Value				
BrowseName	MTConfigurationType				
IsAbstract	True				
References	NodeClass	BrowseName	Data Type	TypeDefinition	Modeling Rule
Subtype of BaseObjectType (See OPCUA Documentation)					

2.1.6 Defintion of MTDeviceType

- 171 The MTDevice is a special type whose object will be the root of the device graph.
 172 The Device uses the component type factory and the component object factories
 173 to create each of the first level components.
- 174 The compositions, relationships, and data items are then recursively created as
 175 one decendes the MTConnect informaiton model.

Table 6: MTDeviceType Definition

Attribute	Value				
BrowseName	MTDeviceType				
IsAbstract	False				
References	NodeClass	BrowseName	Data Type	TypeDefinition	Modeling Rule
Subtype of MTComponentType (see section 2.1.3)					
HasProperty	Variable	Version	String	PropertyType	Optional
HasProperty	Variable	Iso841Class	String	PropertyType	Optional

176 **2.1.6.1 Operations**

- 177 • `deriveBrowseName(element)`
178 Specification: `self.name`
- 179 • `deriveNodeId(element)`
180 Specification: `self.uuid`

181 **2.1.6.2 Constraints**

- 182 • **Constraint `uuid_not_empty`:**
 `uuid->notEmpty()`
183 Documentation: The UUID SHALL be provided.
- 184 • **Constraint `name_not_empty`:**
 `name->notEmpty()`
185 Documentation: The name of the Device SHALL be given.

2.1.7 Defintion of `SensorConfigurationType`

- 186 The `SensorConfiguration` browse name will be created as an Object relationship
187 with the parent component.

Table 7: SensorConfigurationType Definition

Attribute	Value				
BrowseName	SensorConfigurationType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modeling Rule
Subtype of MTConfigurationType (see section 2.1.5)					
HasProperty	Variable	FirmwareVersion	String	PropertyType	Mandatory
HasProperty	Variable	CalibrationDate	UtcTime	PropertyType	Optional
HasProperty	Variable	NextCalibrationDate	UtcTime	PropertyType	Optional
HasProperty	Variable	CalibrationInitials	String	PropertyType	Optional
Organizes	Object	Channels	ChannelType	FolderType	Optional

2.1.8 Defintion of {Component} Type

Table 8: {Component} Type Definition

Attribute	Value				
BrowseName	ComponentType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modeling Rule
Subtype of MTComponentType (see section 2.1.3)					

2.1.9 Defintion of {Composition}Type

Table 9: {Composition}Type Definition

Attribute	Value				
BrowseName	CompositionType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modeling Rule
Subtype of MTCompositionType (see section 2.1.4)					

September 29, 2018

2.2 Data Items

Items.png Items.png

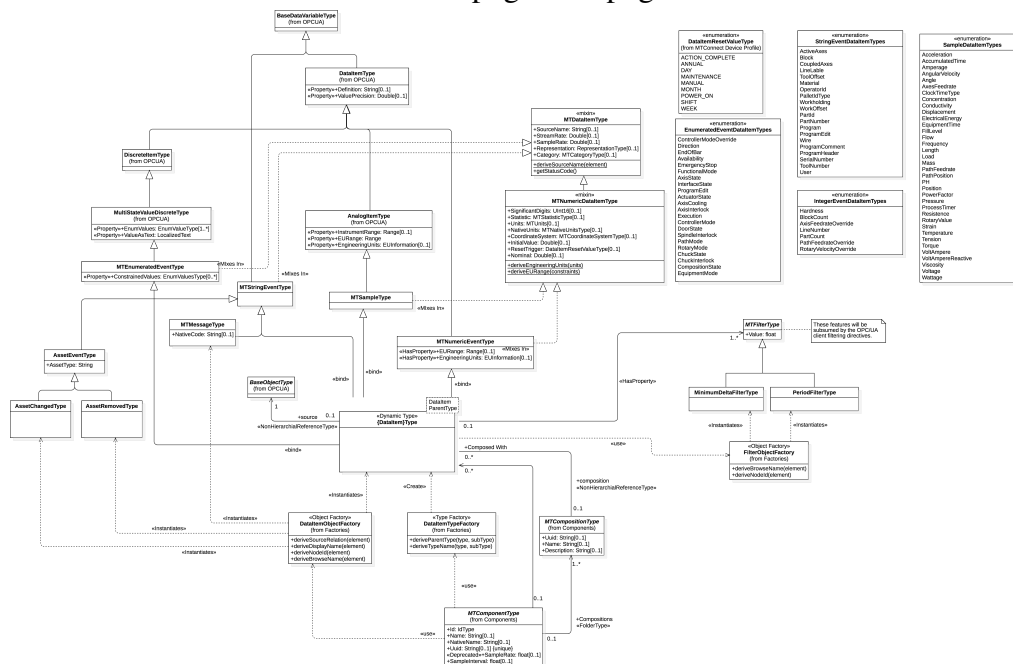


Figure 2: Data Items Diagram

2.2.1 Defintion of AssetChangedType

Table 10: AssetChangedType Definition

Attribute	Value				
BrowseName	AssetChangedType				
IsAbstract	False				
References	NodeClass	BrowseName	Data Type	TypeDefinition	Modeling Rule
Subtype of AssetEventType (see section 2.2.2)					

2.2.2 Defintion of AssetEventType

Table 11: AssetEventType Definition

Attribute	Value				
BrowseName	AssetEventType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modeling Rule
Subtype of MTStringEventType (see section 2.2.11)					
HasProperty	Variable	AssetType	String	PropertyType	Mandatory

2.2.3 Defintion of AssetRemovedType

Table 12: AssetRemovedType Definition

Attribute	Value				
BrowseName	AssetRemovedType				
IsAbstract	False				
References	NodeClass	BrowseName	Data Type	TypeDefinition	Modeling Rule
Subtype of AssetEventType (see section 2.2.2)					

2.2.4 Defintion of MTDataItemType

188 The data item mixin will inject the properties and the methods into the related
189 classes. This facility is similar to the Ruby module mixin or the Scala traits.

Table 13: MTDataItemType Definition

Attribute	Value				
BrowseName	MTDataItemType				
IsAbstract	False				
References	NodeClass	BrowseName	Data Type	TypeDefinition	Modeling Rule
HasProperty	Variable	SourceName	String	PropertyType	Optional
HasProperty	Variable	StreamRate	Double	PropertyType	Optional
HasProperty	Variable	SampleRate	Double	PropertyType	Optional
HasProperty	Variable	Representation	RepresentationType	PropertyType	Optional
HasProperty	Variable	Category	MTCategoryType	PropertyType	Mandatory
HasProperty	Variable	<Dynamic>	MTFilterType	<Dynamic>	Optional
HasComponent	Object	source		BaseObjectType	Optional

2.2.4.1 Operations

- `deriveSourceName(element)`

Specification: `self.Source.CDATA`

Documentation: Derive the source name from the Source element CDATA.

This will represent the alternative long name for the data item's source.

- `getStatusCode()`

Documentation: The OPC/UA status code will be created using the following process:

- If the value of the data item is UNAVAILABLE a status code of `Uncertain_NoCommunicationLastUsable`
- When a reset trigger is specified, new `Good_` status codes will be created. See `ResetTrigger` enumeration.

2.2.5 Defintion of MTEnumeratedEventType

All Data Items with Category EVENT having a Controlled Vocabularies will be of this type. Otherwise, `MTString`

Table 14: MTEnumeratedEventType Definition

Attribute	Value				
BrowseName	MTEnumeratedEventType				
IsAbstract	False				
References	NodeClass	BrowseName	Data Type	TypeDefinition	Modeling Rule
Subtype of MultiStateValueDiscreteType (See OPCUA Documentation)					
HasProperty	Variable	ConstrainedValues	EnumValuesType	PropertyType	Mandatory

2.2.6 Defintion of MTFilterType

204 These features will be subsumed by the OPC/UA client filtering directives.

Table 15: MTFilterType Definition

Attribute	Value				
BrowseName	MTFilterType				
IsAbstract	True				
References	NodeClass	BrowseName	Data Type	TypeDefinition	Modeling Rule
HasProperty	Variable	Value	float	PropertyType	Mandatory

205 2.2.6.1 Operations

- 206 • `deriveBrowseName(element)`
 207 **Specification:** `concat(parent.BrowseName, pascalCase(element.type))`
- 208 • `deriveNodeId(element)`
 209 **Specification:** `concat(parent.NodeId, pascalCase(element.type))`

2.2.7 Defintion of MTMessageType

Table 16: MTMessageType Definition

Attribute	Value				
BrowseName	MTMessageType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modeling Rule
Subtype of MTStringEventType (see section 2.2.11)					
HasProperty	Variable	NativeCode	String	PropertyType	Optional

2.2.8 Defintion of MTNumericDataItemType

210 These are the additional attributes that are relevent to numeric data items. The
211 factory will evaluate these values and will set the engineering units and the range
212 associated with the parent entity.

Table 17: MTNumericDataItemType Definition

Attribute	Value				
BrowseName	MTNumericDataItemType				
IsAbstract	False				
References	NodeClass	BrowseName	Data Type	TypeDefinition	Modeling Rule
Subtype of MTDataItemType (see section 2.2.4)					
HasProperty	Variable	SignificantDigits	UInt16	PropertyType	Optional
HasProperty	Variable	Statistic	MTStatisticType	PropertyType	Optional
HasProperty	Variable	Units	MTUnits	PropertyType	Optional
HasProperty	Variable	NativeUnits	MTNativeUnitsType	PropertyType	Optional
HasProperty	Variable	CoordinateSystem	MTCoordinateSystemType	PropertyType	Optional
HasProperty	Variable	InitialValue	Double	PropertyType	Optional
HasProperty	Variable	ResetTrigger	DataItemResetValueType	PropertyType	Optional
HasProperty	Variable	Nominal	Double	PropertyType	Optional

2.2.8.1 Operations

- deriveEngineeringUnits(units)
 Specification: `EngineeringUnits <- self.units`
- deriveEURange(constraints)
 Specification: `EURange.Low <- self.Constraints.Minimum` `EURange.High <- self.Constraints.Maximum`
 Documentation: Uses the MTConnect Constraints element if present to derive the minimum and maximum values for the numeric values. This applies to both the Numeric Event and the Sample types.

2.2.9 Defintion of MTNumericEventType

- All data items with category EVENT and a numeric value.

Table 18: MTNumericEventType Definition

Attribute	Value				
BrowseName	MTNumericEventType				
IsAbstract	False				
References	NodeClass	BrowseName	Data Type	TypeDefinition	Modeling Rule
Subtype of DataItemType (See OPCUA Documentation)					
HasProperty	Variable	EURange	Range	PropertyType	Optional
HasProperty	Variable	EngineeringUnits	EUInformation	PropertyType	Optional

2.2.10 Defintion of MTSampleType

223 Data Items with category SAMPLE

Table 19: MTSampleType Definition

Attribute	Value				
BrowseName	MTSampleType				
IsAbstract	False				
References	NodeClass	BrowseName	Data Type	TypeDefinition	Modeling Rule
Subtype of AnalogItemType (See OPCUA Documentation)					

2.2.11 Defintion of MTStringEventType

- 224 All data items with category EVENT where the data is freeform text. The set_
 225 data_type constraint derives makes the data type a string for this type.

Table 20: MTStringEventType Definition

Attribute	Value				
BrowseName	MTStringEventType				
IsAbstract	False				
References	NodeClass	BrowseName	Data Type	TypeDefinition	Modeling Rule
Subtype of BaseDataVariableType (See OPCUA Documentation)					

2.2.11.1 Constraints

- Constraint `set_data_type`:

```
derive: DataType <-String
```

2.2.12 Defintion of MinimumDeltaFilterType

Table 21: MinimumDeltaFilterType Definition

Attribute	Value				
BrowseName	MinimumDeltaFilterType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modeling Rule
Subtype of MFilterType (see section 2.2.6)					

2.2.13 Defintion of PeriodFilterType

Table 22: PeriodFilterType Definition

Attribute	Value				
BrowseName	PeriodFilterType				
IsAbstract	False				
References	NodeClass	BrowseName	Data Type	TypeDefinition	Modeling Rule
Subtype of MFilterType (see section 2.2.6)					

2.2.14 Definition of {DataItem}Type

228 For each DataItem the Sub Type, and the Type will be composed to be the HasTypeDefinition
 229 relationship of the object. The BrowseName will also include the Composition
 230 Type if a composition Id is provided.

Table 23: {DataItem}Type Definition

Attribute	Value				
BrowseName	DataItemType				
IsAbstract	False				
References	NodeClass	BrowseName	Data Type	TypeDefinition	Modeling Rule
Subtype of MTNumericEventType (see section 2.2.9)					

September 29, 2018

2.3 Conditions

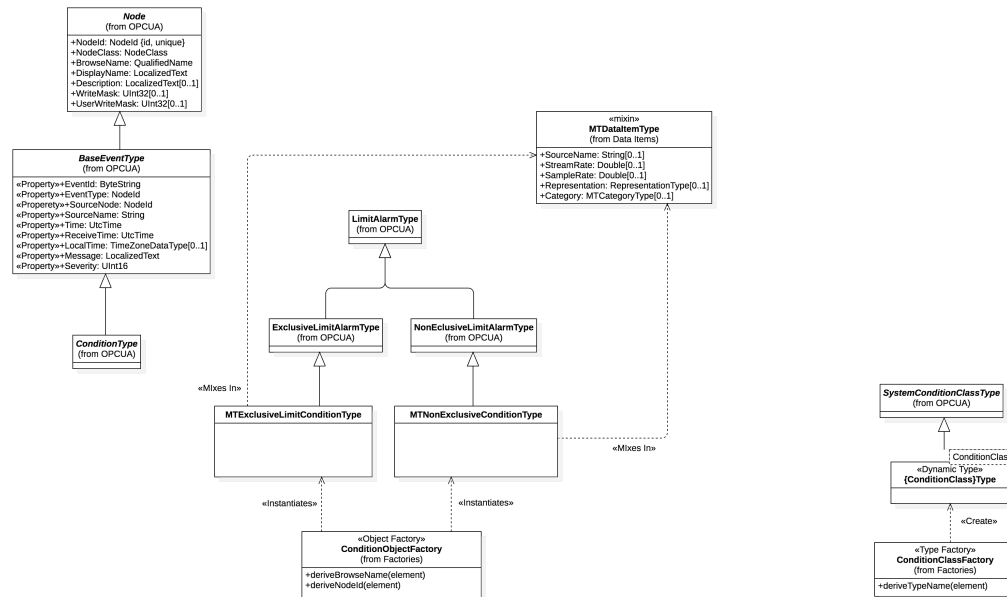


Figure 3: Conditions Diagram

2.3.1 Defintion of MTEXclusiveLimitConditionType

Table 24: MTEXclusiveLimitConditionType Definition

Attribute	Value				
BrowseName	MTExclusiveLimitConditionType				
IsAbstract	False				
References	NodeClass	BrowseName	Data Type	TypeDefinition	Modeling Rule
Subtype of ExclusiveLimitAlarmType (See OPCUA Documentation)					

2.3.2 Defintion of MTNonExclusiveConditionType

Table 25: MTNonExclusiveConditionType Definition

Attribute	Value				
BrowseName	MTNonExclusiveConditionType				
IsAbstract	False				
References	NodeClass	BrowseName	Data Type	TypeDefinition	Modeling Rule
Subtype of NonEclusiveLimitAlarmType (See OPCUA Documentation)					

2.3.3 Defintion of {ConditionClass}Type

Table 26: {ConditionClass}Type Definition

Attribute	Value				
BrowseName	ConditionClassType				
IsAbstract	False				
References	NodeClass	BrowseName	Data Type	TypeDefinition	Modeling Rule
Subtype of SystemConditionClassType (See OPCUA Documentation)					

2.4 Factories

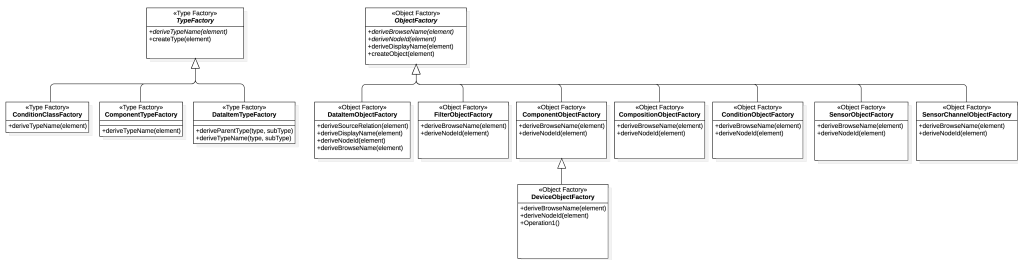


Figure 4: Factories Diagram

231 The factories are not part of the OPC/UA information model. They are a set
 232 of helper classes that are used to create dynamic types and objects. Since the
 233 MTConnect information model can be layered on top of the OPC/UA abstractions,
 234 the factories provide the rules for creating the browse and display names for each
 235 type.

236 The factories also create dynamic objects when required for variables of various
 237 classes when they are required, such as the Data Items and the Components. Some
 238 of the relationships are more complex since they require a dynamic super-type
 239 relationship that relies on the correct placement of the MTConnect elements to be
 240 correctly represented using the OPC/UA base types.

241 This is especially evident when mapping the DataItems and the Conditions to the
 242 MTConnect Information Models and providing sufficient definition to allow for
 243 unambiguous implementation.

2.4.1 Defintion of «Object Factory» `ComponentObjectFactory`

244 2.4.1.1 Operations

245 • `deriveBrowseName(element)`
 246 Specification: `concat(element.QName, (if self.name.notEmpty()`
 247 `then concat('[' , self.name, ']')) else " endif))`

248 • `deriveNodeId(element)`
 249 Specification: `concat(self.findDevice().uuid, element.id)`

2.4.2 Defintion of «Type Factory» `ComponentTypeFactory`

250 The ‘`ComponentTypeFactory`’ creates component types using the MTConnect
 251 XML element as an input. The factory takes the ‘`QName`’ (or qualified name)
 252 of the XML element and then appends ‘`Type`’. For example an ‘`<Controller`
 253 `id=’...’></...>`’ element will create an OPC/UA ‘`ControllerType`’ type definition
 254 as an extension of the base ‘`MTControllerType`’.

255 Currently there is no additional abstractions or super types required by the com-
 256 panion specification. The types will be a single level where each Component is a
 257 sub-type of the base 'MTComponentType'.

258 **2.4.2.1 Operations**

259 • `deriveTypeName(element)`
 260 Specification: `derive: Component <- element.QName`
 261 Documentation: The QName of the element for the component will be used
 262 to derive the type of the node.

2.4.3 Defintion of «Object Factory» CompositionObjectFactory

263 **2.4.3.1 Operations**

264 • `deriveBrowseName(element)`
 265 Specification: `concat(pascalCase(element.type), (if self.name.notEmpty()`
 266 `then concat('[' , self.name, ']')) else " endif))`
 267 • `deriveNodeId(element)`
 268 Specification: `concat(self.findDevice().uuid, element.id)`

2.4.4 Defintion of «Type Factory» CompositionTypeFactory

269 **2.4.4.1 Operations**

270 • `deriveTypeName(element)`
 271 Specification: `derive: Composition <- pascalCase(element.type)`
 272 Documentation: The type for the composition will be created using the pas-
 273 cal case of the 'type' from the composition element.

2.4.5 Defintion of «Type Factory» `ConditionClassFactory`

274 2.4.5.1 Operations

- 275 • `deriveTypeName(element)`
276 Documentation: Create condition classes based on the OPC/UA three con-
277 dition types.

2.4.6 Defintion of «Object Factory» `ConditionObjectFactory`

278 2.4.6.1 Operations

- 279 • `deriveBrowseName(element)`
- 280 • `deriveNodeId(element)`

2.4.7 Defintion of «Object Factory» `DataItemObjectFactory`

281 2.4.7.1 Operations

- 282 • `deriveSourceRelation(element)`
283 Documentation: Use the source composition, component id, or data item id
284 to locate the source node id for this relationship. If one exists, add an object
285 with browse name "source" that relates to the entity referenced by the id.
286 The most specific identity should be used in the following order:
 - 287 – `DataItemId`
 - 288 – `CompositionId`
 - 289 – `ComponentId`
- 290 Since the data item implies composition and component and the compo-
291 sition implies component, there should only be one attribute given for the
292 source.

293 • `deriveDisplayName(element)`
294 Documentation: Same as the `BrowseName`.

295 • `deriveNodeId(element)`
296 Documentation: The `nodeId` will be given by the device `uuid` and the `DataItem`
297 `id` attribute.

298 • `deriveBrowseName(element)`
299 Documentation: The browse name will be composed of the following parts
300 of the model:

301 1. If the `compositionId` is present, the `compositionId` will be resolved the
302 the `Composition` element and the pascal case of the type attribute will
303 be placed first.

304 2. If the `subType` is present, the pascal case of the `subType` will be placed
305 next.

306 3. The pascal case of the type will be placed last.

307 For example, for a data item with the following attributes:

308 – type: `TEMPERATURE`
309 – composition type: `STORAGE_BATTERY`

310 will have the following browse name: `StorageBatteryTemperature`

311 For the data item with the following attributes:

312 – type: `ANGLE`
313 – subType: `ACTUAL`
314 – composition type: `ENCODER`

315 will have the following browse name: `EncoderActualAngle`

2.4.8 Defintion of «Type Factory» `DataItemTypeFactory`

316 Based on the data item category, type, and `subType`, this class creates a new
317 OPC/UA type and also provides the template parameter for the `ParentType` from
318 which this type is derived.

319 2.4.8.1 Operations

- 320 • `deriveParentType(type, subType)`
 321 Documentation: The parent type is derived from the category as follows:
 - 322 – `SAMPLE -> SampleType`
 - 323 – `EVENT ->`
 - 324 * `Enumerated Value -> MTEnumeratedEventType`
 - 325 * `Integer Value -> MTNumericEventType`
 - 326 * `Otherwise -> MTStringEventType`
- 327 • `deriveTypeName(type, subType)`
 328 Specification: `concat(pascalCase(subType), pascalCase(type))`
 329 Documentation: Used to derive the class name for creating a pascal case
 330 name from the sub type and the type. For example type `ROTARY_VELOCITY`
 331 and subType `ACTUAL` will become `ActualRotaryVelocity`.

2.4.9 Defintion of «Object Factory» DeviceObjectFactory

332 The model instantiation for `MTConnect` begins with the ‘Device’ `MTConnect`
 333 element and then recursively traverses the sub-elements. The device will the ca-
 334 pabilities in the component factory to generate all the data items and component
 335 types.

336 2.4.9.1 Operations

- 337 • `deriveBrowseName(element)`
 338 Specification: `derive: element.name`
- 339 • `deriveNodeId(element)`
 340 Specification: `derive: element.uuid`

2.4.10 Defintion of «Object Factory» **FilterObjectFactory**

341 Creates filters based on the type attribute of the Filter element.

342 2.4.10.1 Operations

- 343 • `deriveBrowseName(element)`
- 344 • `deriveNodeId(element)`
- 345 Documentation: The node id is composed of the data item id and the browse
- 346 name.

2.4.11 Defintion of «Object Factory» **ObjectFactory**

347 2.4.11.1 Operations

- 348 • `deriveBrowseName(element)`
- 349 • `deriveNodeId(element)`
- 350 • `deriveDisplayName(element)`
- 351 Specification: `deriveBrowseName(element)`
- 352 • `createObject(element)`

2.4.12 Defintion of «Object Factory» **SensorChannelObjectFactory**

353 2.4.12.1 Operations

- 354 • `deriveBrowseName(element)`
- 355 Specification: `concat('Channel', self.number)`
- 356 • `deriveNodeId(element)`
- 357 Specification: `concat(self.parent.NodeId, BrowseName)`

2.4.13 Defintion of «Object Factory» SensorObjectFactory

358 2.4.13.1 Operations

- 359 • `deriveBrowseName(element)`
360 Specification: `element.QName`
- 361 • `deriveNodeId(element)`
362 Specification: `concat(self.parent.NodeId, BrowseName)`

2.4.14 Defintion of «Type Factory» TypeFactory

363 2.4.14.1 Operations

- 364 • `deriveTypeName(element)`
- 365 • `createType(element)`

2.5 MTConnect Device Profile

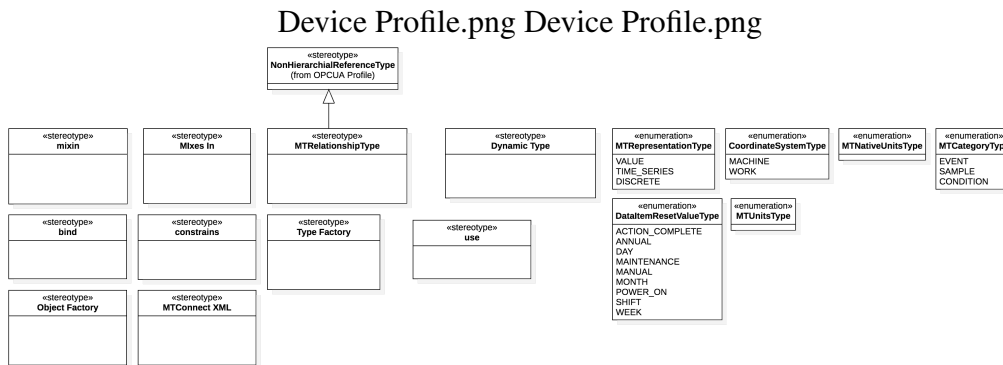


Figure 5: MTConnect Device Profile Diagram

366 The device profile documents the common data types and stereotypes that are used
 367 to construct the model. A stereotype is a design or modeling pattern that provides
 368 additional information about the type or the relationship between types.

369 It can also identify the behavior of a property or the role the type or relation will
 370 play in the model.

371 Stereotypes are used throughout the model to provide additional information that
 372 will help provide context and definition to aid in better understanding the data
 373 model.

2.5.1 Defintion of Dynamic Type

2.5.2 Defintion of MTConnect XML

2.5.3 Defintion of MTRelationshipType

2.5.4 Defintion of Mixes In

374 This stereotype is associated with the dependency between a type and a mixin.
 375 See Section 2.5.9 for a complete description of the mixin.

2.5.5 Defintion of Object Factory

2.5.6 Defintion of Type Factory

2.5.7 Defintion of bind

376 When a dynamic type (See Section 2.5.1) creates an instance where the super-type
377 can be associated based on the data item category and type, the `Type Factory`
378 will specify which supertype is to be referenced.

379 The `bind` stereotype indicates the relationship between the dynamic sub-type and
380 the parent type are resolved baed on the `MTConnect DataItem` meta data.

2.5.8 Defintion of constrains

2.5.9 Defintion of mixin

381 The mixin pattern injects the properties and operations into the types that are
382 related to the using the `Mixes In` dependency. Mixins allow for lightweight
383 multiple inheritance. Since OPC/UA does not allow for multiple inheritance and
384 the `MTConnect` types require the same set of properties when they are sub-typed
385 from existing OPC/UA types, this mechanism allows for this relationship to be
386 expressed.

2.5.10 Defintion of use

387 The `use` stereotype indicates that one class uses as a helper to perform a specific
388 operation or activity. This stereotype is mainly used to indicate that a specific
389 factory is being employed by another type to create dynamic properties or rela-
390 tionships.