



**MTConnect® Standard**  
**Guide: 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<sup>®</sup> Specification and Materials

2

3 AMT – The Association For Manufacturing Technology (“AMT”) owns the copy-  
4 right in this MTConnect<sup>®</sup> 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<sup>®</sup> Specification or Ma-  
7 terial, provided that you may only copy or redistribute the MTConnect<sup>®</sup> 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<sup>®</sup> Specification or Material.

11 If you intend to adopt or implement an MTConnect<sup>®</sup> Specification or Material  
12 in a product, whether hardware, software or firmware, which complies with an  
13 MTConnect<sup>®</sup> Specification, you **SHALL** agree to the MTConnect<sup>®</sup> Specifica-  
14 tion Implementer License Agreement (“Implementer License”) or to the MTConnect<sup>®</sup>  
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<sup>®</sup> Implementers to adopt or implement the MTConnect<sup>®</sup> Specifica-  
18 tions, including certain license rights covering necessary patent claims for that  
19 purpose. These materials can be found at [www.MTConnect.org](http://www.MTConnect.org), or by contact-  
20 ing Paul Warndorf at [mailto:pwarndorf@mtconnect.hyperoffice.](mailto:pwarndorf@mtconnect.hyperoffice.com)  
21 [com](http://www.MTConnect.org).

22 MTConnect<sup>®</sup> 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<sup>®</sup> 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<sup>®</sup>  
28 Institute have any obligation to secure any such rights.

29 This Material and all MTConnect<sup>®</sup> Specifications and Materials are provided “as  
30 is” and MTConnect<sup>®</sup> 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<sup>®</sup>  
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<sup>®</sup> Institute or AMT be liable to  
37 any user or implementer of MTConnect<sup>®</sup> 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<sup>®</sup> Specification or other  
42 MTConnect<sup>®</sup> Materials, whether or not they had advance notice of the possibility  
43 of such damage.

## 44 Table of Contents

45	<b>1 Introduction</b>	<b>1</b>
46	1.1 Overview . . . . .	1
47	<b>2 Types</b>	<b>1</b>
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	2.1.4 Defintion of MTCompositionType . . . . .	6
54	2.1.5 Defintion of MTConfigurationType . . . . .	7
55	2.1.6 Defintion of MTDeviceType . . . . .	8
56	2.1.6.1 Operations . . . . .	9
57	2.1.7 Defintion of SensorConfigurationType . . . . .	9
58	2.1.8 Defintion of ComponentType . . . . .	10
59	2.1.9 Defintion of CompositionType . . . . .	11
60	2.2 Data Items . . . . .	12
61	2.2.1 Defintion of AssetChangedType . . . . .	14
62	2.2.2 Defintion of AssetEventType . . . . .	15
63	2.2.3 Defintion of AssetRemovedType . . . . .	16
64	2.2.4 Defintion of MTDataItemType . . . . .	17
65	2.2.4.1 Operations . . . . .	18
66	2.2.5 Defintion of MTEnumeratedEventType . . . . .	18
67	2.2.6 Defintion of MTFilterType . . . . .	19
68	2.2.6.1 Operations . . . . .	20
69	2.2.7 Defintion of MTMessageType . . . . .	20
70	2.2.8 Defintion of MTNumericDataItemType . . . . .	21
71	2.2.8.1 Operations . . . . .	22
72	2.2.9 Defintion of MTNumericEventType . . . . .	22
73	2.2.10 Defintion of MTSampleType . . . . .	23
74	2.2.11 Defintion of MTStringEventType . . . . .	24
75	2.2.12 Defintion of MinimumDeltaFilterType . . . . .	25
76	2.2.13 Defintion of PeriodFilterType . . . . .	26
77	2.2.14 Defintion of DataItemType . . . . .	27
78	2.3 Conditions . . . . .	28
79	2.3.1 Defintion of MTExclusiveLimitConditionType . . . . .	29

80	2.3.2	Defintion of MTNonExclusiveConditionType . . . . .	30
81	2.3.3	Defintion of ConditionClassType . . . . .	31
82	2.4	Factories . . . . .	32
83	2.4.1	Defintion of ComponentObjectFactory . . . . .	33
84	2.4.1.1	Operations . . . . .	33
85	2.4.2	Defintion of ComponentTypeFactory . . . . .	33
86	2.4.2.1	Operations . . . . .	34
87	2.4.3	Defintion of CompositionObjectFactory . . . . .	34
88	2.4.3.1	Operations . . . . .	34
89	2.4.4	Defintion of CompositionTypeFactory . . . . .	34
90	2.4.4.1	Operations . . . . .	34
91	2.4.5	Defintion of ConditionClassFactory . . . . .	34
92	2.4.5.1	Operations . . . . .	34
93	2.4.6	Defintion of ConditionObjectFactory . . . . .	35
94	2.4.6.1	Operations . . . . .	35
95	2.4.7	Defintion of DataItemObjectFactory . . . . .	35
96	2.4.7.1	Operations . . . . .	35
97	2.4.8	Defintion of DataItemTypeFactory . . . . .	35
98	2.4.8.1	Operations . . . . .	35
99	2.4.9	Defintion of DeviceObjectFactory . . . . .	36
100	2.4.9.1	Operations . . . . .	36
101	2.4.10	Defintion of FilterObjectFactory . . . . .	36
102	2.4.10.1	Operations . . . . .	36
103	2.4.11	Defintion of ObjectFactory . . . . .	36
104	2.4.11.1	Operations . . . . .	36
105	2.4.12	Defintion of SensorChannelObjectFactory . . . . .	37
106	2.4.12.1	Operations . . . . .	37
107	2.4.13	Defintion of SensorObjectFactory . . . . .	37
108	2.4.13.1	Operations . . . . .	37
109	2.4.14	Defintion of TypeFactory . . . . .	37
110	2.4.14.1	Operations . . . . .	37
111	2.5	MTConnect Device Profile . . . . .	38
112	2.5.1	Defintion of Dynamic Type . . . . .	38
113	2.5.2	Defintion of Mixes In . . . . .	38
114	2.5.3	Defintion of MTConnect XML . . . . .	38
115	2.5.4	Defintion of MTRelationshipType . . . . .	38
116	2.5.5	Defintion of Object Factory . . . . .	39
117	2.5.6	Defintion of Type Factory . . . . .	39

118	2.5.7	Defintion of bind . . . . .	39
119	2.5.8	Defintion of constrains . . . . .	39
120	2.5.9	Defintion of mixin . . . . .	39
121	2.5.10	Defintion of use . . . . .	39

122 **List of Figures**

123 **Figure 1: Components Diagram** . . . . . 2

124 **Figure 2: Data Items Diagram** . . . . . 13

125 **Figure 3: Conditions Diagram** . . . . . 29

126 **Figure 4: Factories Diagram** . . . . . 32

127 **Figure 5: MTConnect Device Profile Diagram** . . . . . 38

# 1 Introduction

128 The following conventions will be used throughout the document to provide a  
129 clear and consistent understanding of the use of each type of data and information  
130 used to define the MTConnect<sup>®</sup> standard and associated data.

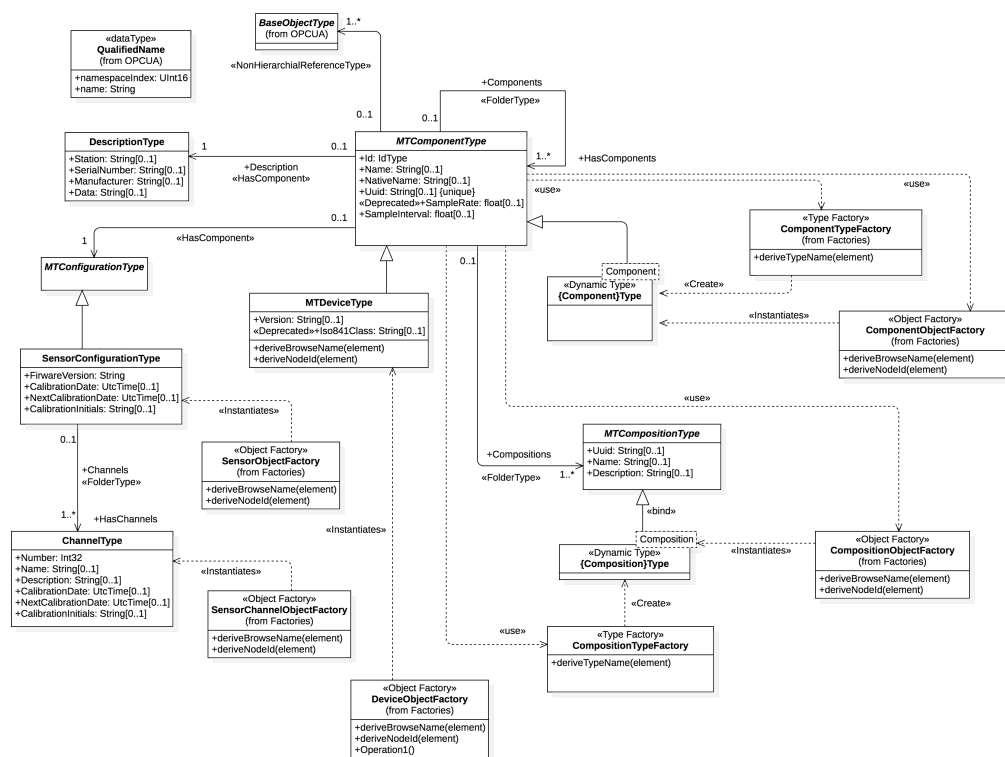
## 1.1 Overview

131 Overview of the standards...

# 2 Types

## 2.1 Components





### Figure 1: Components Diagram

September 29, 2018

132 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

- 133 The desription provides some general information about the manufacture and se-  
 134 rial number of the component. In the XML, the CDATA is freeform text that is  
 135 represented in the Data Property of the Description Object.

**Table 2:** DescriptionType Definition

Attribute	Value				
BrowseName	DescriptionType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	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

### 2.1.2.1 Operations

- deriveBrowseName(element)

Specification: "Description"

- deriveNodeId(element)

Specification: `concat(self.parent.NodeId, BrowseName)`

### 2.1.3 Defintion of MTComponentType

The base Component Type from which all MTConnect Components are derived from. The component type factory is used to create the specific OPC/UA types as subtypes of the MTConnect 'MTComponentType'. The component types will be created once for all Component objects of that type based on the 'QName' of the MTConnect XML element.

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

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

**Table 3:** MTComponentType Definition

Attribute	Value				
BrowseName	MTComponentType				
IsAbstract	True				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modeling Rule
HasProperty	Variable	Id	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>	DataItemType	<Dynamic>	Optional
HasProperty	Variable	<Dynamic>	BaseObjectType	<Dynamic>	Optional
Organizes	Object	Conditions	MTNonExclusiveConditionType	FolderType	Optional
HasProperty	Variable	<Dynamic>	DataItemType	<Dynamic>	Mandatory

151 The OPC/UA Object with browse name ‘Linear[X]’ will be created with the  
 152 HasTypeDefinition referencing the ‘Linear’ OPC/UA type.

153 The meta data for the component and it’s relationships are static. The dynamic  
 154 data will be represented using the \_OPC/UA Part 8\_

## 2.1.4 Defintion of MTCompositionType

**Table 4:** MTCompositionType Definition

Attribute	Value				
BrowseName	MTCompositionType				
IsAbstract	True				
References	NodeClass	BrowseName	DataType	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
NonHierarchicalReferenceType	Object	ecomposition	DataItemType	NonHierarchicalReferenceType	Optional

## 2.1.5 Defintion of MTConfigurationType

**Table 5:** MTConfigurationType Definition

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

## 2.1.6 Defintion of MTDeviceType

- 155 The MTDevice is a special type whose object will be the root of the device graph.  
 156 The Device uses the component type factory and the component object factories  
 157 to create each of the first level components.
- 158 The compositions, relationships, and data items are then recursively created as  
 159 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

#### 160 **2.1.6.1 Operations**

- 161     • `deriveBrowseName(element)`  
162         Specification: `self.name`
- 163     • `deriveNodeId(element)`  
164         Specification: `self.uuid`

### **2.1.7 Defintion of SensorConfigurationType**

165 The SensorConfiguration browse name will be created as an Object relationship  
166 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 ComponentType

**Table 8:** ComponentType 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 CompositionType

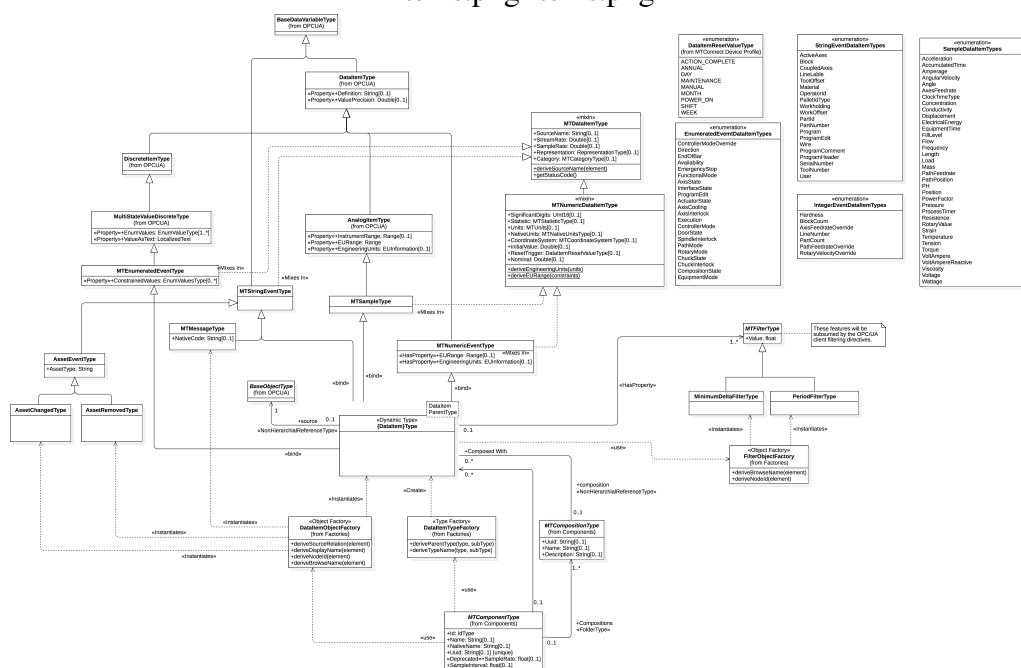
**Table 9:** CompositionType Definition

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	DataType	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	DataType	TypeDefinition	Modeling Rule
Subtype of AssetEventType (see section 2.2.2)					

#### **2.2.4 Defintion of MTDataItemType**

167 The data item mixin will inject the properties and the methods into the related  
168 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	Optional
HasProperty	Variable	<Dynamic>	MTFilterType	<Dynamic>	Optional
HasComponent	Object	source		BaseObjectType	Optional

#### 169 2.2.4.1 Operations

- 170 • deriveSourceName(element)  
 171 Specification: `self.Source.CDATA`  
 172 Documentation: Derive the source name from the Source element CDATA.  
 173 This will represent the alternative long name for the data item's source.
- 174 • getStatusCode()  
 175 Documentation: The OPC/UA status code will be created using the follow-  
 176 ing process:  
 177 \* If the value of the data item is 'UNAVAILABLE' a status code of 'Un-  
 178 certain\_NoCommunicationLastUsable'. \* When a reset trigger is specified,  
 179 new 'Good\_' status codes will be created. See 'ResetTrigger' enumeration.

### 2.2.5 Defintion of MTEnumeratedEventType

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

**Table 14:** MTEnumeratedEventType Definition

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

## 2.2.6 Defintion of MTFilterType

182 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	DataType	TypeDefinition	Modeling Rule
HasProperty	Variable	Value	float	PropertyType	Mandatory

### 183 2.2.6.1 Operations

- 184     • `deriveBrowseName(element)`  
 185         Specification: `concat (parent.BrowseName, pascalCase(element.type))`
- 186     • `deriveNodeId(element)`  
 187         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**

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

**Table 17:** MTNumericDataItemType Definition

Attribute	Value				
BrowseName	MTNumericDataItemType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	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	DataType	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

201 Data Items with category SAMPLE

**Table 19:** MTSampleType Definition

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

### 2.2.11 Definition of MTStringEventType

202 All data items with category EVENT where the data is freeform text. The set\_  
 203 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	DataType	TypeDefinition	Modeling Rule
Subtype of BaseDataVariableType (See OPCUA Documentation)					

## 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 MTFilterType (see section 2.2.6)					



### 2.2.13 Defintion of PeriodFilterType

**Table 22:** PeriodFilterType Definition

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

## 2.2.14 Definition of DataItemType

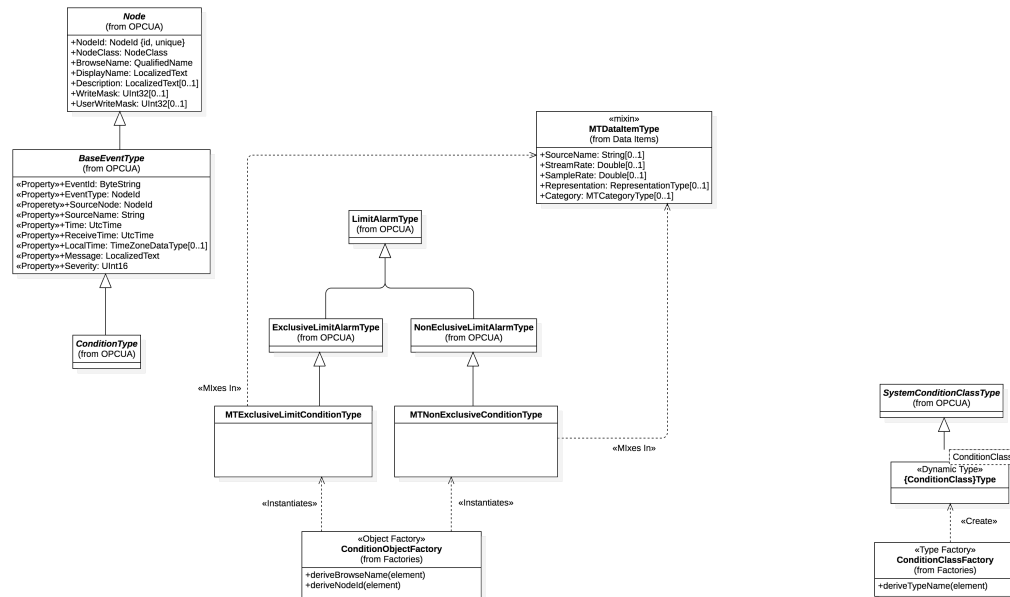
204 For each DataItem the Sub Type, and the Type will be composed to be the HasType-  
 205 Definition relationship of the object. The BrowseName will also include the Com-  
 206 position Type if a composition Id is provided.

**Table 23:** DataItemType Definition

Attribute	Value				
BrowseName	DataItemType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	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 MTEExclusiveLimitConditionType

### Table 24: MTEExclusiveLimitConditionType Definition

<b>Attribute</b>	<b>Value</b>				
BrowseName	MTExclusiveLimitConditionType				
IsAbstract	False				
<b>References</b>	<b>NodeClass</b>	<b>BrowseName</b>	<b>Data Type</b>	<b>Type Definition</b>	<b>Modeling Rule</b>
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	DataType	TypeDefinition	Modeling Rule
Subtype of NonEclusiveLimitAlarmType (See OPCUA Documentation)					

### 2.3.3 Defintion of ConditionClassType

**Table 26:** ConditionClassType Definition

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

## 2.4 Factories

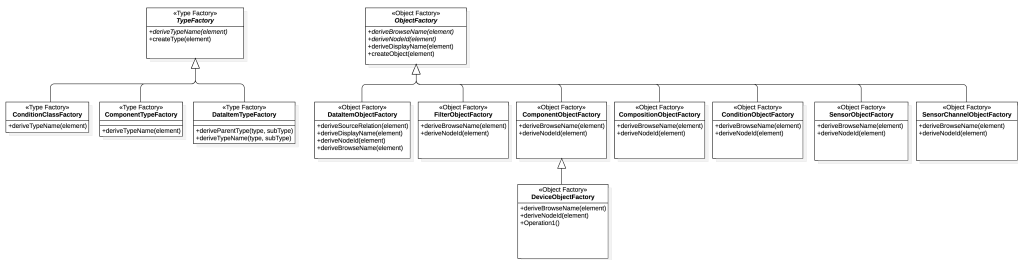


Figure 4: Factories Diagram

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

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

217 This is especially evident when mapping the DataItems and the Conditions to the  
 218 MTConnect Information Models and providing sufficient definition to allow for  
 219 unambiguous implementation.

## 2.4.1 Defintion of ComponentObjectFactory

### 220 2.4.1.1 Operations

- 221 • deriveBrowseName(element)  
 222     Specification: `concat(element.QName, (if self.name.notEmpty()  
 223     then concat('[' , self.name, ']')) else " endif))`
- 224 • deriveNodeId(element)  
 225     Specification: `concat(self.findDevice().uuid, element.id)`

## 2.4.2 Defintion of ComponentTypeFactory

226 The 'ComponentTypeFactory' creates component types using the MTConnect  
 227 XML element as an input. The factory takes the 'QName' (or qualified name)  
 228 of the XML element and then appends 'Type'. For example an '<Controller  
 229 id='...'></...>' element will create an OPC/UA 'ControllerType' type definition  
 230 as an extension of the base 'MTControllerType'.



231 Currently there is no additional abstractions or super types required by the com-  
 232 panion specification. The types will be a single level where each Component is a  
 233 sub-type of the base ‘MTComponentType’.

#### 234 **2.4.2.1 Operations**

- 235 • deriveTypeName(element)
- 236     Specification: `derive: Component <- element.QName`
- 237     Documentation: The QName of the element for the component will be used
- 238     to derive the type of the node.

### **2.4.3 Defintion of CompositionObjectFactory**

#### 239 **2.4.3.1 Operations**

- 240 • deriveBrowseName(element)
- 241     Specification: `concat(pascalCase(element.type), (if self.name.notEmpty()`
- 242     `then concat('[' , self.name, ']')) else " endif))`
- 243 • deriveNodeId(element)
- 244     Specification: `concat(self.findDevice().uuid, element.id)`

### **2.4.4 Defintion of CompositionTypeFactory**

#### 245 **2.4.4.1 Operations**

- 246 • deriveTypeName(element)
- 247     Specification: `derive: Composition <- pascalCase(element.type)`
- 248     Documentation: The type for the composition will be created using the pas-
- 249     cal case of the ‘type’ from the composition element.

## 2.4.5 Defintion of ConditionClassFactory

### 250 2.4.5.1 Operations

- 251 • deriveTypeName(element)  
252 Documentation: Create condition classes based on the OPC/UA three con-  
253 dition types.

## 2.4.6 Defintion of ConditionObjectFactory

### 254 2.4.6.1 Operations

- 255 • deriveBrowseName(element)
- 256 • deriveNodeId(element)

## 2.4.7 Defintion of DataItemObjectFactory

### 257 2.4.7.1 Operations

- 258 • deriveSourceRelation(element)  
259 Documentation: Use the source composition, component id, or data item id  
260 to locate the source node id for this relationship. If one exists, add an object  
261 with browse name "source" that relates to the entity referenced by the id.  
262 The most specific identity should be used in the following order: \* DataItemId  
263 \* CompositionId \* ComponentId  
264 Since the data item implies composition and component and the compo-  
265 sition implies component, there should only be one attribute given for the  
266 source.
- 267 • deriveDisplayName(element)  
268 Documentation: Same as the BrowseName.

- 269 • deriveNodeId(element)
- 270     Documentation: The nodeId will be given by the device uuid and the DataItem
- 271     id attribute.
- 272 • deriveBrowseName(element)
- 273     Documentation: The browse name will be composed of the following parts
- 274     of the model:
- 275     1. If the compositionId is present, the compositionId will be resolved the
- 276     the Composition element and the pascal case of the type attribute will be
- 277     placed first. 2. If the subType is present, the pascal case of the subType will
- 278     be placed next. 3. The pascal case of the type will be placed last.
- 279     For example, for a data item with the following attributes:
- 280     \* type: TEMPERATURE \* composition type: STORAGE\_BATTERY
- 281     will have the browseName "StorageBatteryTemperature"
- 282     For the data item with the following attributes:
- 283     \* type: ANGLE \* subType: ACTUAL \* composition type: ENCODER
- 284     will have the following browse name: "EncoderActualAngle"

## 2.4.8 Defintion of DataItemTypeFactory

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

288 See the Data Item Type Factory.

### 289 2.4.8.1 Operations

- 290 • deriveParentType(type, subType)
- 291     Documentation: The parent type is derived from the category as follows:
- 292     \* SAMPLE -> SampleType \* EVENT -> \* Enumerated Value -> MTEnum-
- 293     eratedEventType \* Integer Value -> MTNumericEventType \* Otherwise
- 294     -> MTStringEventType

- 295 • `deriveTypeName(type, subType)`
- 296     Specification: `concat (pascalCase(subType), pascalCase(type))`
- 297     Documentation: Used to derive the class name for creating a pascal case
- 298     name from the sub type and the type. For example type `ROTARY_VE-`
- 299     `LOCITY` and subType `ACTUAL` will become `ActualRotaryVelocity`.

## 2.4.9 Defintion of DeviceObjectFactory

300 The model instantiation for `MTConnect` begins with the ‘Device’ `MTConnect`

301 element and then recursively traverses the sub-elements. The device will the ca-

302 pabilities in the component factory to generate all the data items and component

303 types.

### 304 2.4.9.1 Operations

- 305 • `deriveBrowseName(element)`
- 306     Specification: `derive: element.name`
- 307 • `deriveNodeId(element)`
- 308     Specification: `derive: element.uuid`
- 309 • `Operation1()`

## 2.4.10 Defintion of FilterObjectFactory

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

### 311 2.4.10.1 Operations

- 312 • `deriveBrowseName(element)`
- 313 • `deriveNodeId(element)`
- 314     Documentation: The node id is composed of the data item id and the browse
- 315     name.

## 2.4.11 Defintion of ObjectFactory

### 316 2.4.11.1 Operations

- 317     • deriveBrowseName(element)
- 318     • deriveNodeId(element)
- 319     • deriveDisplayName(element)
- 320         Specification: `deriveBrowseName(element)`
- 321     • createObject(element)

## 2.4.12 Defintion of SensorChannelObjectFactory

### 322 2.4.12.1 Operations

- 323     • deriveBrowseName(element)
- 324         Specification: `concat('Channel', self.number)`
- 325     • deriveNodeId(element)
- 326         Specification: `concat(self.parent.NodeId, BrowseName)`

## 2.4.13 Defintion of SensorObjectFactory

### 327 2.4.13.1 Operations

- 328     • deriveBrowseName(element)
- 329         Specification: `element.QName`
- 330     • deriveNodeId(element)
- 331         Specification: `concat(self.parent.NodeId, BrowseName)`

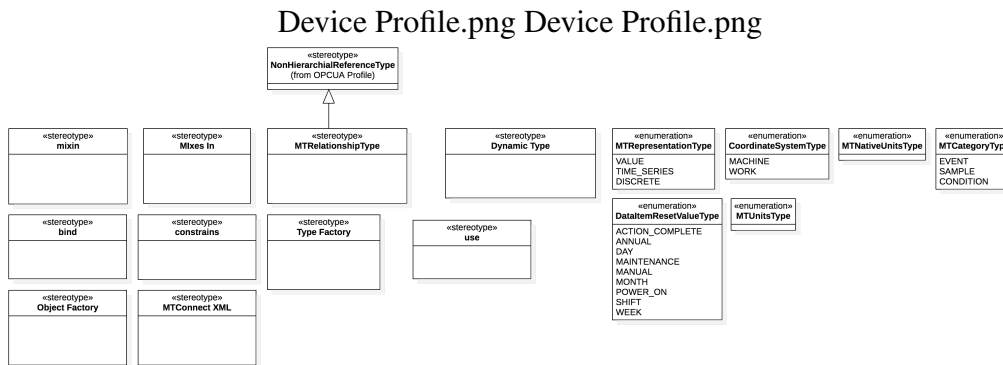


Figure 5: MTConnect Device Profile Diagram

## 2.4.14 Defintion of TypeFactory

### 332 2.4.14.1 Operations

- 333     • deriveTypeName(element)
- 334     • createType(element)

## 2.5 MTConnect Device Profile

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

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

340 Stereotypes are used throughout the model to provide additional information that  
 341 will halp provide context and definition to aid in better understanding the data  
 342 model.

### **2.5.1 Defintion of Dynamic Type**

### **2.5.2 Defintion of MIXes In**

### **2.5.3 Defintion of MTConnect XML**

### **2.5.4 Defintion of MTRelationshipType**

### **2.5.5 Defintion of Object Factory**

### **2.5.6 Defintion of Type Factory**

### **2.5.7 Defintion of bind**

### **2.5.8 Defintion of constrains**

### **2.5.9 Defintion of mixin**

343 The contents properties and the behavior of the class are combined with another  
344 class.

### **2.5.10 Defintion of use**

345 The use stereotype indicates that one class uses as a helper to perform a specific  
346 operation or activity. This stereotype is mainly used to indicate that a specific  
347 factory is being employed by another type to create dynamic properties or rela-  
348 tionships.