



**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® 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	<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 {Component}Type . . . . .	10
59	2.1.9 Defintion of {Composition}Type . . . . .	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 {DataItem}Type . . . . .	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 {ConditionClass}Type . . . . .	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 . . . . .	35
92	2.4.5.1	Operations . . . . .	35
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 . . . . .	36
98	2.4.8.1	Operations . . . . .	37
99	2.4.9	Defintion of DeviceObjectFactory . . . . .	37
100	2.4.9.1	Operations . . . . .	37
101	2.4.10	Defintion of FilterObjectFactory . . . . .	38
102	2.4.10.1	Operations . . . . .	38
103	2.4.11	Defintion of ObjectFactory . . . . .	38
104	2.4.11.1	Operations . . . . .	38
105	2.4.12	Defintion of SensorChannelObjectFactory . . . . .	38
106	2.4.12.1	Operations . . . . .	38
107	2.4.13	Defintion of SensorObjectFactory . . . . .	39
108	2.4.13.1	Operations . . . . .	39
109	2.4.14	Defintion of TypeFactory . . . . .	39
110	2.4.14.1	Operations . . . . .	39
111	2.5	MTConnect Device Profile . . . . .	39
112	2.5.1	Defintion of Dynamic Type . . . . .	40
113	2.5.2	Defintion of MTConnect XML . . . . .	40
114	2.5.3	Defintion of MTRelationshipType . . . . .	40
115	2.5.4	Defintion of Mixes In . . . . .	40
116	2.5.5	Defintion of Object Factory . . . . .	41
117	2.5.6	Defintion of Type Factory . . . . .	41

September 29, 2018

118	2.5.7	Defintion of bind . . . . .	41
119	2.5.8	Defintion of constrains . . . . .	41
120	2.5.9	Defintion of mixin . . . . .	41
121	2.5.10	Defintion of use . . . . .	41

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** . . . . . 40

# 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



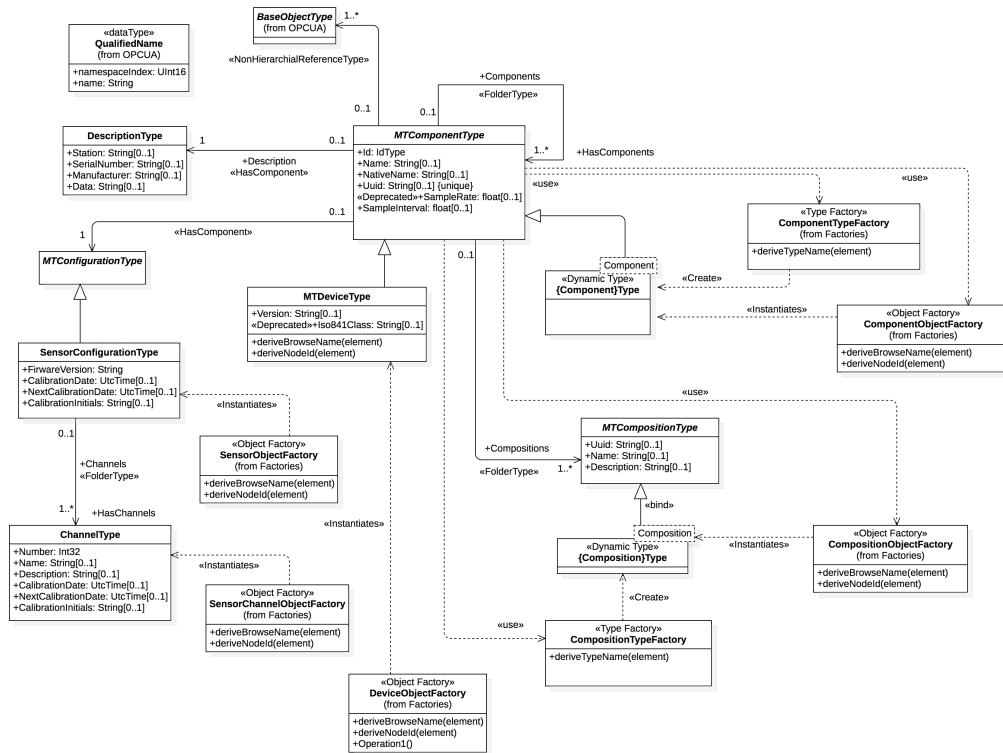


Figure 1: Components Diagram

September 29, 2018

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

### **2.1.1 Defintion of ChannelType**

**Table 1:** `escape_name` 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:** `escape_name` 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

### 136 2.1.2.1 Operations

137     • `deriveBrowseName(element)`

138         Specification: "Description"

139     • `deriveNodeId(element)`

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

## 2.1.3 Defintion of MTComponentType

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

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

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

**Table 3:** escape\_name 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:** escape\_name 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:** escape\_name 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:** `escape_name` 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:** `escape_name` 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:** `escape_name` 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

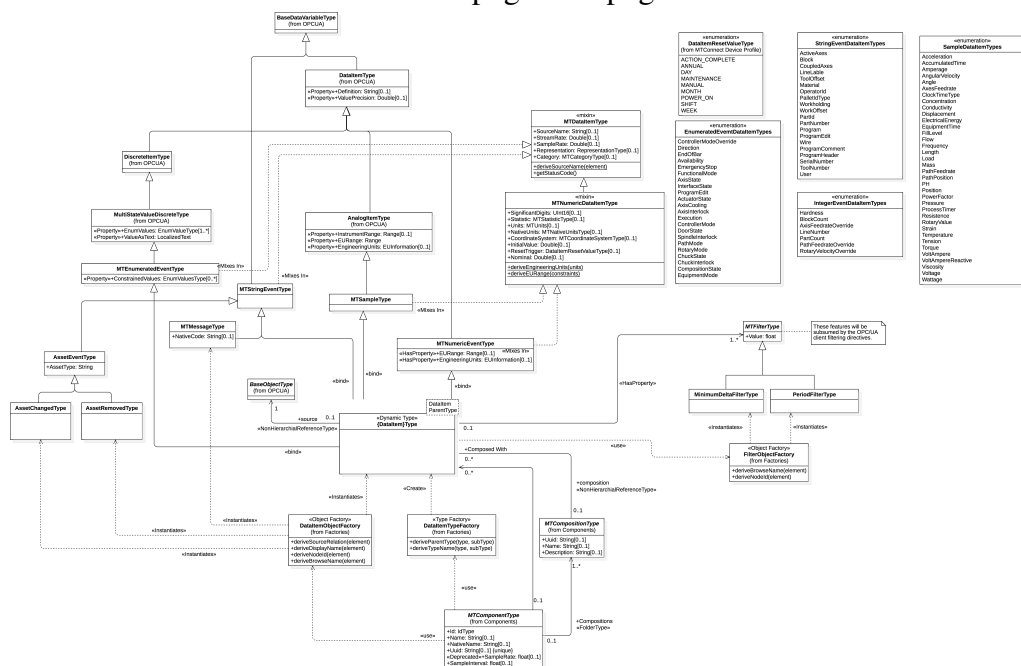
**Table 9:** escape\_name Definition

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:** escape\_name 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:** escape\_name 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:** escape\_name 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:** `escape_name` Definition

Attribute	Value				
BrowseName	MTDataItemType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	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

#### 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 `Uncertain_-`  
178             `NoCommunicationLastUsable`
  - 179         – When a reset trigger is specified, new `Good_` status codes will be  
180             created. See `ResetTrigger` enumeration.

### 2.2.5 Defintion of `MTEnumeratedEventType`

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

**Table 14:** escape\_name 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

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

**Table 15:** escape\_name Definition

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

184 **2.2.6.1 Operations**

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

**2.2.7 Defintion of MTMessageType****Table 16:** `escape_name` 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**

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

**Table 17:** `escape_name` 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:** escape\_name 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

202 Data Items with category SAMPLE

**Table 19:** escape\_name 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

203 All data items with category EVENT where the data is freeform text. The set\_  
 204 data\_type constraint derives makes the data type a string for this type.

**Table 20:** escape\_name 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:** escape\_name 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:** escape\_name 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 {DataItem} Type

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

**Table 23:** escape\_name 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**

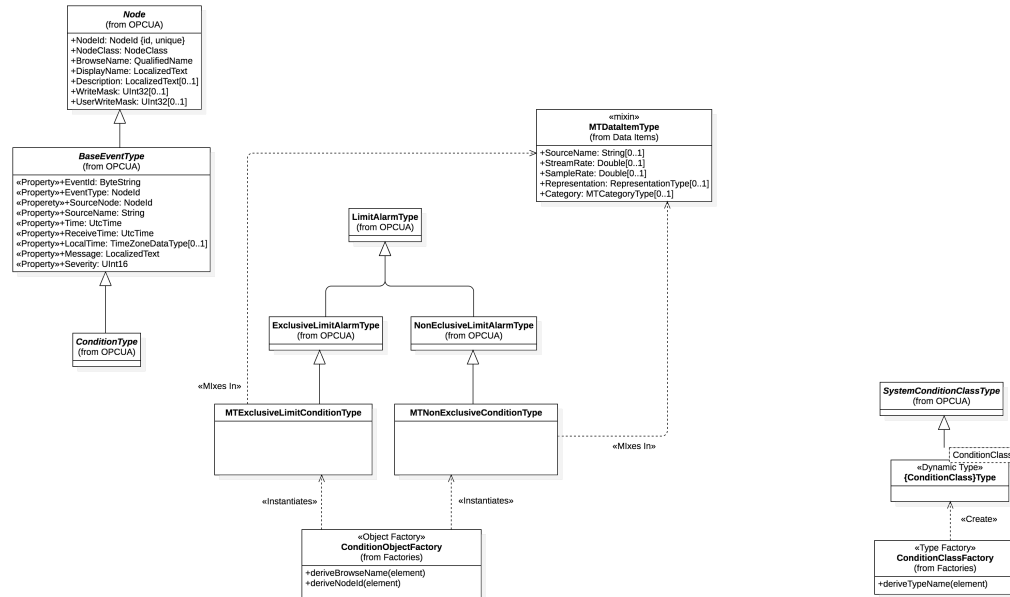


Figure 3: Conditions Diagram

### 2.3.1 Defintion of MTExclusiveLimitConditionType

Table 24: escape\_name Definition

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

### 2.3.2 Defintion of MTNonExclusiveConditionType

**Table 25:** escape\_name 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 {ConditionClass}Type

**Table 26:** escape\_name Definition

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

## 2.4 Factories

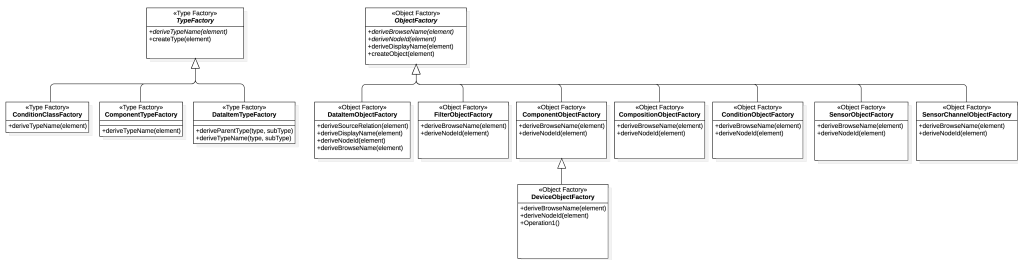


Figure 4: Factories Diagram

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

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

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

## 2.4.1 Defintion of «Object Factory» `ComponentObjectFactory`

### 221 2.4.1.1 Operations

222 • `deriveBrowseName(element)`  
 223     Specification: `concat(element.QName, (if self.name.notEmpty()`  
 224     `then concat('[' , self.name, ']')) else " endif))`

225 • `deriveNodeId(element)`  
 226     Specification: `concat(self.findDevice().uuid, element.id)`

## 2.4.2 Defintion of «Type Factory» `ComponentTypeFactory`

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



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

#### 235 2.4.2.1 Operations

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

### 2.4.3 Defintion of «Object Factory» `CompositionObjectFactory`

#### 240 2.4.3.1 Operations

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

### 2.4.4 Defintion of «Type Factory» `CompositionTypeFactory`

#### 246 2.4.4.1 Operations

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

## 2.4.5 Defintion of «Type Factory» `ConditionClassFactory`

### 251 2.4.5.1 Operations

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

## 2.4.6 Defintion of «Object Factory» `ConditionObjectFactory`

### 255 2.4.6.1 Operations

- 256 • `deriveBrowseName(element)`
- 257 • `deriveNodeId(element)`

## 2.4.7 Defintion of «Object Factory» `DataItemObjectFactory`

### 258 2.4.7.1 Operations

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

270     • `deriveDisplayName(element)`  
271         Documentation: Same as the `BrowseName`.

272     • `deriveNodeId(element)`  
273         Documentation: The `nodeId` will be given by the device `uuid` and the `DataItem`  
274         `id` attribute.

275     • `deriveBrowseName(element)`  
276         Documentation: The browse name will be composed of the following parts  
277         of the model:

278         1. If the `compositionId` is present, the `compositionId` will be resolved the  
279             the `Composition` element and the pascal case of the type attribute will  
280             be placed first.

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

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

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

285         – type: `TEMPERATURE`  
286         – composition type: `STORAGE_BATTERY`

287     will have the following browse name: `StorageBatteryTemperature`

288     For the data item with the following attributes:

289         – type: `ANGLE`  
290         – subType: `ACTUAL`  
291         – composition type: `ENCODER`

292     will have the following browse name: `EncoderActualAngle`

## 2.4.8 Defintion of «Type Factory» `DataItemTypeFactory`

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

### 296 2.4.8.1 Operations

- 297 • `deriveParentType(type, subType)`  
 298 Documentation: The parent type is derived from the category as follows:
  - 299 – `SAMPLE -> SampleType`
  - 300 – `EVENT ->`
    - 301 \* `Enumerated Value -> MTEnumeratedEventType`
    - 302 \* `Integer Value -> MTNumericEventType`
    - 303 \* `Otherwise -> MTStringEventType`
- 304 • `deriveTypeName(type, subType)`  
 305 Specification: `concat(pascalCase(subType), pascalCase(type))`  
 306 Documentation: Used to derive the class name for creating a pascal case  
 307 name from the sub type and the type. For example type `ROTARY_VELOCITY`  
 308 and subType `ACTUAL` will become `ActualRotaryVelocity`.

## 2.4.9 Defintion of «Object Factory» DeviceObjectFactory

309 The model instantiation for MTConnect begins with the ‘Device’ MTConnect  
 310 element and then recursively traverses the sub-elements. The device will the ca-  
 311 pabilities in the component factory to generate all the data items and component  
 312 types.

### 313 2.4.9.1 Operations

- 314 • `deriveBrowseName(element)`  
 315 Specification: `derive: element.name`
- 316 • `deriveNodeId(element)`  
 317 Specification: `derive: element.uuid`

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

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

### 319 2.4.10.1 Operations

- 320 • `deriveBrowseName(element)`
- 321 • `deriveNodeId(element)`
- 322 Documentation: The node id is composed of the data item id and the browse
- 323 name.

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

### 324 2.4.11.1 Operations

- 325 • `deriveBrowseName(element)`
- 326 • `deriveNodeId(element)`
- 327 • `deriveDisplayName(element)`
- 328 Specification: `deriveBrowseName(element)`
- 329 • `createObject(element)`

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

### 330 2.4.12.1 Operations

- 331 • `deriveBrowseName(element)`
- 332 Specification: `concat('Channel', self.number)`
- 333 • `deriveNodeId(element)`
- 334 Specification: `concat(self.parent.NodeId, BrowseName)`

### **2.4.13 Defintion of «Object Factory» SensorObjectFactory**

#### **335 2.4.13.1 Operations**

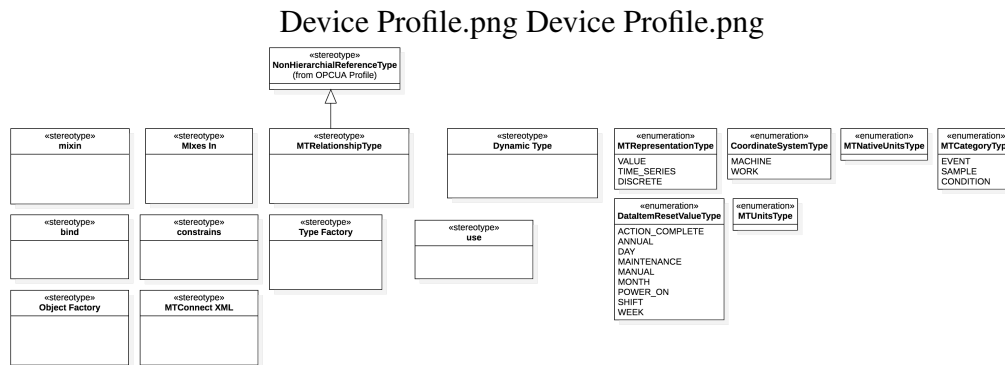
- 336     • `deriveBrowseName(element)`  
337         Specification: `element.QName`
- 338     • `deriveNodeId(element)`  
339         Specification: `concat(self.parent.NodeId, BrowseName)`

### **2.4.14 Defintion of «Type Factory» TypeFactory**

#### **340 2.4.14.1 Operations**

- 341     • `deriveTypeName(element)`
- 342     • `createType(element)`

## **2.5 MTConnect Device Profile**



**Figure 5: MTConnect Device Profile Diagram**

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

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

348 Stereotypes are used throughout the model to provide additional information that  
 349 will help provide context and definition to aid in better understanding the data  
 350 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

351 This stereotype is associated with the dependency between a type and a mixin.  
 352 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**

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

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

## **2.5.8 Defintion of constrains**

## **2.5.9 Defintion of mixin**

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

## **2.5.10 Defintion of use**

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