OPC Unified Architecture

for

MTConnect®

Companion Specification

Release Candidate 2.0RC8

February 19, 2019

Specification Type:	Industry Standard Specification	Comments:	
Title:	OPC Unified Architecture for MTConnect	Date:	February 19, 2019
Version:	2.0RC8	Software:	LaTeX
Authors:	William Sobel, Randy Armstrong, John Turner, Russell Waddell, Shaurabh Singh	Source:	OPC_UA_MTConnect_2.0RC8.pdf
Owner:	MTConnect Institute	Status:	Release Candidate

Document History

Version	Date	Reason	Comments	Mantis
2.00 RC8	2019-02-18	Revision	Added list of tables	4631
2.00 RC8	2019-02-18	Revision	Added table for reset triggers and suggested status codes	4630
2.00 RC8	2019-02-18	Revision	Removed data rate issue from page 77	4629
2.00 RC7	2019-02-12	Revision	2.0 RC6: 8.4.6.2 MTConnect Condition Branching Example – Need to improve documentation of condition branching	4608
2.00 RC7	2019-02-12	Revision	Document 2.0 RC 6: 8.4.7 Messages – need to improve message handling	4607
2.00 RC7	2019-02-12	Node Ids	List of NodeIds as CSV	4612
2.00 RC7	2019-02-12	Revision	A chapter for Profiles and Namespaces needs to be added	4611
2.00.06	2019-01-22	Revision	Added missing class types and made MTMessageType a Variable instead of an Event	
2.00.05	2018-12-09	Revision	Fixed Composition	
2.00.04	2018-11-30	Initial	Initial Release Candidate	

Contents

1 Scope **1**

2 OPC Unified Architecture for MTConnect Companion Specification Goals **2**

3 Who Will Find Benefit from this Companion Specification? **3**

4 Normative References **3**
4.1 OPC UA References 3
4.2 MTConnect References 4
4.3 Other References 4

5 Terms, Definitions and Conventions **4**
5.1 Overview 5
5.2 Conventions 5
5.3 Terms and Acronnyms 5
5.3.1 Conventions for Node descriptions 5
5.3.2 NodeIds and BrowseNames 7
5.3.3 Common Attributes 8

6 Introduction to MTConnect and OPC UA **11**
6.1 MTConnect 11
6.1.1 Data Dictionary 12
6.1.2 Semantic Data Models 12
6.1.3 Fundamentals of MTConnect 13
6.2 Introduction to OPC Unified Architecture 14
6.2.1 Basics of OPC UA 15
6.2.2 Information Modeling in OPC UA 16

7 Use Cases **21**
7.1 Machine Tool Manufacturer with Existing MTConnect Implementation . . 22
7.2 Software Vendor 23
7.3 Data Scientist 24
7.4 Industrial Systems Integrator 25

8 Mapping the MTConnect Information Model to OPC UA **26**
8.1 MTConnect UML Representation of OPC 26
8.1.1 MTConnect UML Model 29
8.2 MTConnect Information Model 30
8.3 Mapping The Model 31
8.3.1 Mapping Rules and Conventions 31
8.3.2 Component and Composition **`BrowseName`** and ***Type*** Rules 33
8.3.3 `DataItem` **`HasTypeDefinition`** and **`BrowseName`** Conventions 34
8.3.4 Mapping `MTDataItem units` to **`EngineeringUnits`** 37
8.3.5 Mapping Example 39

8.4 MTConnect Streaming Data 58
8.4.1 MTConnectStreams Document Header 59
8.4.2 MTConnectStreams Device and Component Stream 59
8.4.3 Samples 60
8.4.4 String and Numeric Events 61
8.4.5 Controlled Vocabulary Events 61
8.4.6 Conditions 62
8.4.7 Messages 67
8.5 Time Series Samples 68

9 MTConnect OPC UA Types **71**
9.1 Components 71
9.1.1 Defintion of `MTChannelType` 71
9.1.2 Defintion of `MTComponentType` 72
9.1.3 Defintion of `MTDeviceType` 75
9.1.4 Defintion of `MTCompositionType` 75
9.1.5 Defintion of `MTConfigurationType` 76
9.1.6 Defintion of `MTSensorConfigurationType` 76
9.1.7 Defintion of `MTDescriptionType` 77
9.2 Component Types 77
9.2.1 Defintion of `ActuatorType` 78
9.2.2 Defintion of `AuxiliariesType` 78
9.2.3 Defintion of `BarFeederType` 79
9.2.4 Defintion of `EnvironmentalType` 79
9.2.5 Defintion of `LoaderType` 79
9.2.6 Defintion of `SensorType` 80
9.2.7 Defintion of `ToolingDeliveryType` 80
9.2.8 Defintion of `WasteDisposalType` 80
9.2.9 Defintion of `AxesType` 81
9.2.10 Defintion of `LinearType` 81
9.2.11 Defintion of `RotaryType` 81
9.2.12 Defintion of `ChuckType` 82
9.2.13 Defintion of `ControllerType` 82
9.2.14 Defintion of `PathType` 82
9.2.15 Defintion of `DoorType` 83
9.2.16 Defintion of `InterfacesType` 83
9.2.17 Defintion of `BarFeederInterfaceType` 83
9.2.18 Defintion of `ChuckInterfaceType` 84
9.2.19 Defintion of `DoorInterfaceType` 84
9.2.20 Defintion of `MaterialHandlerInterfaceType` 84
9.2.21 Defintion of `ResourcesType` 85
9.2.22 Defintion of `MaterialsType` 85
9.2.23 Defintion of `StockType` 85
9.2.24 Defintion of `PersonnelType` 86
9.2.25 Defintion of `SystemsType` 86

9.2.26 Defintion of CoolantType 87
9.2.27 Defintion of DielectricType 87
9.2.28 Defintion of ElectricType 87
9.2.29 Defintion of EnclosureType 87
9.2.30 Defintion of FeederType 88
9.2.31 Defintion of HydraulicType 88
9.2.32 Defintion of LubricationType 88
9.2.33 Defintion of PneumaticType 89
9.2.34 Defintion of ProcessPowerType 89
9.2.35 Defintion of ProtectiveType 89
9.3 Data Items 90
9.3.1 Defintion of AssetEventDataType 91
9.3.2 Defintion of MTAssetEventType 91
9.3.3 Defintion of MTConditionClassType 92
9.3.4 Defintion of MTConstraintType 93
9.3.5 Defintion of MTControlledVocabEventType 93
9.3.6 Defintion of «mixin» MTDataItemType 94
9.3.7 Defintion of «mixin» MTNumericDataItemType 97
9.3.8 Defintion of MTEventClassType 99
9.3.9 Defintion of MTMessageEventType 100
9.3.10 Defintion of MTMessageType 100
9.3.11 Defintion of MTNumericEventType 101
9.3.12 Defintion of MTSampleType 103
9.3.13 Defintion of MTStringEventType 105
9.3.14 Defintion of MTThreeSpaceSampleType 106
9.3.15 Defintion of MessageDataType 108
9.3.16 Defintion of ThreeSpaceSampleDataType 108
9.4 Conditions 108
9.4.1 Defintion of MTConditionType 110
9.5 Data Item Types 112
9.5.1 Defintion of MTDataItemClassType 112
9.5.2 Defintion of MTMessageClassType 113
9.6 Sample Data Item Types 113
9.6.1 Defintion of MTSampleClassType 113
9.6.2 Defintion of LoadClassType 116
9.6.3 Defintion of AccelerationClassType 116
9.6.4 Defintion of AccumulatedTimeClassType 116
9.6.5 Defintion of AngularAccelerationClassType 116
9.6.6 Defintion of AngularVelocityClassType 117
9.6.7 Defintion of AmperageClassType 117
9.6.8 Defintion of AngleClassType 117
9.6.9 Defintion of AxisFeedrateClassType 118
9.6.10 Defintion of ClockTimeClassType 118
9.6.11 Defintion of ConcentrationClassType 118
9.6.12 Defintion of ConductivityClassType 119

9.6.13 Defintion of DisplacementClassType 119
9.6.14 Defintion of ElectricalEnergyClassType 119
9.6.15 Defintion of EquipmentTimerClassType 120
9.6.16 Defintion of FillLevelClassType 120
9.6.17 Defintion of FlowClassType 120
9.6.18 Defintion of FrequencyClassType 121
9.6.19 Defintion of LengthClassType 121
9.6.20 Defintion of LinearForceClassType 121
9.6.21 Defintion of MassClassType 122
9.6.22 Defintion of PathFeedrateClassType 122
9.6.23 Defintion of PathPositionClassType 122
9.6.24 Defintion of PHClassType 123
9.6.25 Defintion of PositionClassType 123
9.6.26 Defintion of PowerFactorClassType 124
9.6.27 Defintion of PressureClassType 124
9.6.28 Defintion of ProcessTimerClassType 124
9.6.29 Defintion of ResistenceClassType 125
9.6.30 Defintion of RotaryVelocityClassType 125
9.6.31 Defintion of SoundLevelClassType 125
9.6.32 Defintion of StrainClassType 126
9.6.33 Defintion of TemperatureClassType 126
9.6.34 Defintion of TensionClassType 126
9.6.35 Defintion of TiltClassType 127
9.6.36 Defintion of TorqueClassType 127
9.6.37 Defintion of VoltAmpereClassType 127
9.6.38 Defintion of VelocityClassType 128
9.6.39 Defintion of VoltAmpereReactiveClassType 128
9.6.40 Defintion of ViscosityClassType 128
9.6.41 Defintion of VoltageClassType 129
9.6.42 Defintion of WattageClassType 129
9.7 Controlled Vocab Data Item Types 129
9.7.1 Defintion of MTControlledVocabEventClassType 130
9.7.2 Defintion of ActuatorStateClassType 131
9.7.3 Defintion of AvailabilityClassType 131
9.7.4 Defintion of AxisCouplingClassType 132
9.7.5 Defintion of AxisInterlockClassType 133
9.7.6 Defintion of AxisStateClassType 133
9.7.7 Defintion of ChuckInterlockClassType 134
9.7.8 Defintion of ChuckStateClassType 135
9.7.9 Defintion of ControllerModeClassType 135
9.7.10 Defintion of ExecutionClassType 136
9.7.11 Defintion of CompositionStateClassType 137
9.7.12 Defintion of ControllerModeOverrideClassType 138
9.7.13 Defintion of DirectionClassType 139
9.7.14 Defintion of DoorStateClassType 139

9.7.15 Defintion of EmergencyStopClassType 140
9.7.16 Defintion of EndOfBarClassType 140
9.7.17 Defintion of EquipmentModeClassType 141
9.7.18 Defintion of FunctionalModeClassType 142
9.7.19 Defintion of SpindleInterlockClassType 142
9.7.20 Defintion of PathModeClassType 143
9.7.21 Defintion of PowerStateClassType 144
9.7.22 Defintion of ProgramEditClassType 144
9.7.23 Defintion of RotaryModeClassType 145
9.7.24 Defintion of InterfaceStateClassType 146
9.7.25 Defintion of MaterialFeedClassType 146
9.7.26 Defintion of MaterialChangeClassType 147
9.7.27 Defintion of MaterialRetractClassType 148
9.7.28 Defintion of MaterialLoadClassType 148
9.7.29 Defintion of MaterialUnloadClassType 149
9.7.30 Defintion of OpenDoorClassType 150
9.7.31 Defintion of CloseDoorClassType 150
9.7.32 Defintion of OpenChuckClassType 151
9.7.33 Defintion of CloseChuckClassType 152
9.7.34 Defintion of PartChangeClassType 152
9.8 Numeric Event Data Item Types 153
9.8.1 Defintion of MTNumericEventClassType 153
9.8.2 Defintion of AxisFeedrateOverrideClassType 154
9.8.3 Defintion of BlockCountClassType 154
9.8.4 Defintion of HardnessClassType 155
9.8.5 Defintion of LineNumberClassType 155
9.8.6 Defintion of PartCountClassType 156
9.8.7 Defintion of RotaryVelocityOverrideClassType 156
9.9 String Event Data Item Types 156
9.9.1 Defintion of MTStringEventClassType 157
9.9.2 Defintion of BlockClassType 158
9.9.3 Defintion of CoupledAxesClassType 158
9.9.4 Defintion of LineLabelClassType 158
9.9.5 Defintion of MaterialClassType 159
9.9.6 Defintion of OperatorIdClassType 159
9.9.7 Defintion of PalletIdClassType 159
9.9.8 Defintion of PartIdClassType 160
9.9.9 Defintion of PartNumberClassType 160
9.9.10 Defintion of ProgramClassType 160
9.9.11 Defintion of ProgramEditNameClassType 161
9.9.12 Defintion of ProgramHeaderClassType 161
9.9.13 Defintion of ProgramCommentClassType 161
9.9.14 Defintion of SerialNumberClassType 162
9.9.15 Defintion of ToolAssetIdClassType 162
9.9.16 Defintion of ToolNumberClassType 162

9.9.17 Defintion of ToolOffsetClassType 163
9.9.18 Defintion of UserClassType 163
9.9.19 Defintion of WireClassType 164
9.9.20 Defintion of WorkholdingClassType 164
9.9.21 Defintion of WorkOffsetClassType 164
9.9.22 Defintion of MessageClassType 165
9.9.23 Defintion of AssetChangedClassType 165
9.9.24 Defintion of AssetRemovedClassType 165
9.9.25 Defintion of LineClassType 165
9.10 Condition Data Item Types 166
9.10.1 Defintion of ActuatorClassType 166
9.10.2 Defintion of CommunicationsClassType 166
9.10.3 Defintion of DataRangeClassType 166
9.10.4 Defintion of HardwareClassType 167
9.10.5 Defintion of LogicProgramClassType 167
9.10.6 Defintion of MotionProgramClassType 167
9.10.7 Defintion of SystemClassType 168
9.11 Data Item Sub Types 168
9.11.1 Defintion of MTDataItemSubClassType 168
9.11.2 Defintion of AbsoluteSubClassType 172
9.11.3 Defintion of ActualSubClassType 172
9.11.4 Defintion of ActionSubClassType 172
9.11.5 Defintion of AllSubClassType 172
9.11.6 Defintion of AlternatingSubClassType 173
9.11.7 Defintion of AScaleSubClassType 173
9.11.8 Defintion of AuxiliarySubClassType 173
9.11.9 Defintion of BadSubClassType 174
9.11.10 Defintion of BrinellSubClassType 174
9.11.11 Defintion of BScaleSubClassType 174
9.11.12 Defintion of CommandedSubClassType 175
9.11.13 Defintion of GoodSubClassType 175
9.11.14 Defintion of ControlSubClassType 175
9.11.15 Defintion of CScaleSubClassType 176
9.11.16 Defintion of DelaySubClassType 176
9.11.17 Defintion of DirectSubClassType 176
9.11.18 Defintion of DryRunSubClassType 177
9.11.19 Defintion of DScaleSubClassType 177
9.11.20 Defintion of FixtureSubClassType 177
9.11.21 Defintion of IncrementalSubClassType 178
9.11.22 Defintion of JobSubClassType 178
9.11.23 Defintion of KineticSubClassType 178
9.11.24 Defintion of LateralSubClassType 179
9.11.25 Defintion of LeebSubClassType 179
9.11.26 Defintion of LengthSubClassType 179
9.11.27 Defintion of LinearSubClassType 180

9.11.28 Defintion of LineSubClassType 180
9.11.29 Defintion of LoadedSubClassType 180
9.11.30 Defintion of MachineAxisLockSubClassType 181
9.11.31 Defintion of MaintenanceSubClassType 181
9.11.32 Defintion of ManualUnclampSubClassType 181
9.11.33 Defintion of MaximumSubClassType 182
9.11.34 Defintion of MinimumSubClassType 182
9.11.35 Defintion of MohsSubClassType 182
9.11.36 Defintion of MoleSubClassType 183
9.11.37 Defintion of MotionSubClassType 183
9.11.38 Defintion of NoScaleSubClassType 183
9.11.39 Defintion of OperatingSubClassType 184
9.11.40 Defintion of OperatorSubClassType 184
9.11.41 Defintion of OptionalStopSubClassType 184
9.11.42 Defintion of OverrideSubClassType 185
9.11.43 Defintion of PrimarySubClassType 185
9.11.44 Defintion of PoweredSubClassType 185
9.11.45 Defintion of ProbeSubClassType 186
9.11.46 Defintion of ProcessSubClassType 186
9.11.47 Defintion of ProgrammedSubClassType 186
9.11.48 Defintion of RadialSubClassType 187
9.11.49 Defintion of RapidSubClassType 187
9.11.50 Defintion of RelativeSubClassType 187
9.11.51 Defintion of RemainingSubClassType 188
9.11.52 Defintion of RequestSubClassType 188
9.11.53 Defintion of ResponseSubClassType 188
9.11.54 Defintion of RockwellSubClassType 189
9.11.55 Defintion of RotarySubClassType 189
9.11.56 Defintion of SetUpSubClassType 189
9.11.57 Defintion of ShoreSubClassType 190
9.11.58 Defintion of StandardSubClassType 190
9.11.59 Defintion of SwitchedSubClassType 190
9.11.60 Defintion of TargetSubClassType 191
9.11.61 Defintion of ToolChangeStopSubClassType 191
9.11.62 Defintion of ToolEdgeSubClassType 191
9.11.63 Defintion of ToolGroupSubClassType 192
9.11.64 Defintion of ToolSubClassType 192
9.11.65 Defintion of UasbleSubClassType 192
9.11.66 Defintion of VerticalSubClassType 193
9.11.67 Defintion of VolumeSubClassType 193
9.11.68 Defintion of VickersSubClassType 193
9.11.69 Defintion of WeightSubClassType 194
9.11.70 Defintion of WorkingSubClassType 194
9.11.71 Defintion of WorkpieceSubClassType 194
9.12 MTConnect Device Profile 195

9.12.1 Defintion of «Deprecated» 195
9.12.2 Defintion of «Dynamic Type» 196
9.12.3 Defintion of «HasMTClassType» 196
9.12.4 Defintion of «HasMTComposition» 196
9.12.5 Defintion of «HasMTReference» 196
9.12.6 Defintion of «HasMTSource» 197
9.12.7 Defintion of «HasMTSubClassType» 197
9.12.8 Defintion of «Mixes In» 197
9.12.9 Defintion of «bind» 198
9.12.10 Defintion of «mixin» 198
9.12.11 Defintion of «use» 198
9.12.12 Defintion of «values» 198

10 Profiles and Namespaces **199**
10.1 Namespace Metadata 199
10.2 Conformance Units and Profiles 199
10.2.1 Server 199
10.2.2 Client 200
10.3 Handling of OPC UA Namespaces 200

Annex A MTConnect Namespace and Mappings (normative) **202**
A.1 Namespace and identifiers for MTConnect Information Model 202

List of Figures

Figure 1: MTConnect Architecture Overview . . . 14
Figure 2: The Scope of OPC UA within an Enterprise . . . 16
Figure 3: A Basic Object in an OPC UA Address Space . . . 17
Figure 4: The Relationship between Type Definitions and Instances . . . 18
Figure 5: Reference Types from other Reference Types . . . 19
Figure 6: References showing hierarchies and/or relationships . . . 19
Figure 7: The OPC UA Information Model Notation for Types and Instances 20
Figure 8: The OPC UA Standard References . . . 20
Figure 9: The Device Manufacturer Use Case . . . 22
Figure 10: Device Manufacturer with Native MTConnect Agent . . . 22
Figure 11: Device Manufacturer with Separate MTConnect Agent . . . 23
Figure 12: The Independent Software Vendor (ISV) Use Case . . . 24
Figure 13: MTConnect OPC UA and ROS Device Integration . . . 25
Figure 14: MTConnect MTComponentType in OPC UA . . . 27
Figure 15: MTConnect MTComponentType in UML . . . 28
Figure 16: MTConnect Example Object . . . 29
Figure 17: MTConnect Device Object . . . 40
Figure 18: MTConnect Device Object with Data Items . . . 42
Figure 19: MTConnect Data Item References . . . 43
Figure 20: Linear X Axis Example . . . 44
Figure 21: Rotary[C] Axis RotaryMode DataItem . . . 46
Figure 22: Rotary[C] Axis RotaryVelocity DataItem . . . 47
Figure 23: Rotary[C] Axis Load DataItem . . . 48
Figure 24: Rotary[C] Axis Motor Amperage DataItem . . . 49
Figure 25: Controller Component and Data Items . . . 51
Figure 26: Path Component and Data Items . . . 52
Figure 27: Electric System Component First Set . . . 54
Figure 28: Electric System Component Second Set . . . 56
Figure 29: Coolant System . . . 58
Figure 30: Condition Branching . . . 64
Figure 31: Components Diagram . . . 71
Figure 32: MTComponentType Diagram . . . 73
Figure 33: Component Types Diagram . . . 78
Figure 34: Data Items Diagram . . . 90
Figure 35: Conditions Diagram . . . 109
Figure 36: MTConnect Device Profile Diagram . . . 195

List of Tables

Table 1: Examples of DataTypes . . . 6
Table 2: Type Definition Table . . . 7
Table 3: Common Node Attributes . . . 9
Table 4: Common Object Attributes . . . 9
Table 5: Common Variable Attributes . . . 10
Table 6: Common VariableTypes Attributes . . . 10
Table 7: Common Method Attributes . . . 11
Table 8: `ExecutionDataType` Enumeration . . . 32
Table 9: `EngineeringUnits` DataType structure . . . 37
Table 10: `EngineeringUnits` DataType structure . . . 38
Table 11: `EngineeringUnits` DataType structure (Continued) . . . 39
Table 12: MTConnect `ResetTrigger` to OPC UA `StatusCode` mapping . . . 55
Table 13: `ControllerModeDataType` Enumeration . . . 62
Table 14: `LogicProgramCondition` States . . . 65
Table 15: `MTChannelType` Definition . . . 72
Table 16: `MTComponentType` Definition . . . 74
Table 17: `MTDeviceType` Definition . . . 75
Table 18: `MTCompositionType` Definition . . . 76
Table 19: `MTConfigurationType` Definition . . . 76
Table 20: `MTSensorConfigurationType` Definition . . . 77
Table 21: `MTDescriptionType` Definition . . . 77
Table 22: `ActuatorType` Definition . . . 78
Table 23: `AuxiliariesType` Definition . . . 79
Table 24: `BarFeederType` Definition . . . 79
Table 25: `EnvironmentalType` Definition . . . 79
Table 26: `LoaderType` Definition . . . 80
Table 27: `SensorType` Definition . . . 80
Table 28: `ToolingDeliveryType` Definition . . . 80
Table 29: `WasteDisposalType` Definition . . . 81
Table 30: `AxesType` Definition . . . 81
Table 31: `LinearType` Definition . . . 81
Table 32: `RotaryType` Definition . . . 82
Table 33: `ChuckType` Definition . . . 82
Table 34: `ControllerType` Definition . . . 82
Table 35: `PathType` Definition . . . 83
Table 36: `DoorType` Definition . . . 83
Table 37: `InterfacesType` Definition . . . 83
Table 38: `BarFeederInterfaceType` Definition . . . 84
Table 39: `ChuckInterfaceType` Definition . . . 84
Table 40: `DoorInterfaceType` Definition . . . 84
Table 41: `MaterialHandlerInterfaceType` Definition . . . 85
Table 42: `ResourcesType` Definition . . . 85
Table 43: `MaterialsType` Definition . . . 85

Table 44: `StockType` Definition 86
Table 45: `PersonnelType` Definition 86
Table 46: `SystemsType` Definition 86
Table 47: `CoolantType` Definition 87
Table 48: `DielectricType` Definition 87
Table 49: `ElectricType` Definition 87
Table 50: `EnclosureType` Definition 88
Table 51: `FeederType` Definition 88
Table 52: `HydraulicType` Definition 88
Table 53: `LubricationType` Definition 89
Table 54: `PneumaticType` Definition 89
Table 55: `ProcessPowerType` Definition 89
Table 56: `ProtectiveType` Definition 90
Table 57: `AssetEventDataType` DataType 91
Table 58: `MTAssetEventType` Definition 92
Table 59: `MTConditionClassType` Definition 93
Table 60: `MTConstraintType` Definition 93
Table 61: `MTControlledVocabEventType` Definition 94
Table 62: `MTDataItemType` Definition 95
Table 63: `MTCategoryType` Enumeration 95
Table 64: `MTRepresentationType` Enumeration 96
Table 65: `MTRepresentationType` Enumeration 96
Table 66: `MTCategoryType` Enumeration 96
Table 67: `MTRepresentationType` Enumeration 97
Table 68: `MTCategoryType` Enumeration 97
Table 69: `MTNumericDataItemType` Definition 98
Table 70: `MTStatisticType` Enumeration 98
Table 71: `MTCoordinateSystemType` Enumeration 99
Table 72: `MTResetTriggerType` Enumeration 99
Table 73: `MTEventClassType` Definition 99
Table 74: `MTMessageEventType` Definition 100
Table 75: `MTMessageType` Definition 101
Table 76: `MTNumericEventType` Definition 102
Table 77: `MTSampleType` Definition 104
Table 78: `MTStringEventType` Definition 106
Table 79: `MTThreeSpaceSampleType` Definition 107
Table 80: `MessageDataType` DataType 108
Table 81: `ThreeSpaceSampleDataType` DataType 108
Table 82: `MTConditionType` Definition 111
Table 83: `MTSeverityDataType` Enumeration 112
Table 84: `QualifierDataType` Enumeration 112
Table 85: `MTDataItemClassType` Definition 113
Table 86: `MTMessageClassType` Definition 113
Table 87: `MTSampleClassType` Definition 114
Table 88: `LoadClassType` Definition 116

Table 89: `AccelerationClassType` Definition . . . 116
Table 90: `AccumulatedTimeClassType` Definition . . . 116
Table 91: `AngularAccelerationClassType` Definition . . . 117
Table 92: `AngularVelocityClassType` Definition . . . 117
Table 93: `AmperageClassType` Definition . . . 117
Table 94: `AngleClassType` Definition . . . 118
Table 95: `AxisFeedrateClassType` Definition . . . 118
Table 96: `ClockTimeClassType` Definition . . . 118
Table 97: `ConcentrationClassType` Definition . . . 119
Table 98: `ConductivityClassType` Definition . . . 119
Table 99: `DisplacementClassType` Definition . . . 119
Table 100: `ElectricalEnergyClassType` Definition . . . 120
Table 101: `EquipmentTimerClassType` Definition . . . 120
Table 102: `FillLevelClassType` Definition . . . 120
Table 103: `FlowClassType` Definition . . . 121
Table 104: `FrequencyClassType` Definition . . . 121
Table 105: `LengthClassType` Definition . . . 121
Table 106: `LinearForceClassType` Definition . . . 122
Table 107: `MassClassType` Definition . . . 122
Table 108: `PathFeedrateClassType` Definition . . . 122
Table 109: `PathPositionClassType` Definition . . . 123
Table 110: `PHClassType` Definition . . . 123
Table 111: `PositionClassType` Definition . . . 123
Table 112: `PowerFactorClassType` Definition . . . 124
Table 113: `PressureClassType` Definition . . . 124
Table 114: `ProcessTimerClassType` Definition . . . 125
Table 115: `ResistenceClassType` Definition . . . 125
Table 116: `RotaryVelocityClassType` Definition . . . 125
Table 117: `SoundLevelClassType` Definition . . . 126
Table 118: `StrainClassType` Definition . . . 126
Table 119: `TemperatureClassType` Definition . . . 126
Table 120: `TensionClassType` Definition . . . 127
Table 121: `TiltClassType` Definition . . . 127
Table 122: `TorqueClassType` Definition . . . 127
Table 123: `VoltAmpereClassType` Definition . . . 128
Table 124: `VelocityClassType` Definition . . . 128
Table 125: `VoltAmpereReactiveClassType` Definition . . . 128
Table 126: `ViscosityClassType` Definition . . . 129
Table 127: `VoltageClassType` Definition . . . 129
Table 128: `WattageClassType` Definition . . . 129
Table 129: `MTControlledVocabEventClassType` Definition . . . 130
Table 130: `ActuatorStateClassType` Definition . . . 131
Table 131: `ActiveStateDataType` Enumeration . . . 131
Table 132: `AvailabilityClassType` Definition . . . 132
Table 133: `AvailabilityDataType` Enumeration . . . 132

Table 134: `AxisCouplingClassType` Definition 132
Table 135: `AxisCouplingDataType` Enumeration 133
Table 136: `AxisInterlockClassType` Definition 133
Table 137: `ActiveStateDataType` Enumeration 133
Table 138: `AxisStateClassType` Definition 134
Table 139: `AxisStateDataType` Enumeration 134
Table 140: `ChuckInterlockClassType` Definition 134
Table 141: `ActiveStateDataType` Enumeration 135
Table 142: `ChuckStateClassType` Definition 135
Table 143: `OpenStateDataType` Enumeration 135
Table 144: `ControllerModeClassType` Definition 136
Table 145: `ControllerModeDataType` Enumeration 136
Table 146: `ExecutionClassType` Definition 136
Table 147: `ExecutionDataType` Enumeration 137
Table 148: `CompositionStateClassType` Definition 137
Table 149: `CompositionStateDataType` Enumeration 138
Table 150: `ControllerModeOverrideClassType` Definition 138
Table 151: `OnOffDataType` Enumeration 138
Table 152: `DirectionClassType` Definition 139
Table 153: `DirectionDataType` Enumeration 139
Table 154: `DoorStateClassType` Definition 139
Table 155: `OpenStateDataType` Enumeration 140
Table 156: `EmergencyStopClassType` Definition 140
Table 157: `EmergencyStopDataType` Enumeration 140
Table 158: `EndOfBarClassType` Definition 141
Table 159: `YesNoDataType` Enumeration 141
Table 160: `EquipmentModeClassType` Definition 141
Table 161: `OnOffDataType` Enumeration 142
Table 162: `FunctionalModeClassType` Definition 142
Table 163: `FunctionalModeDataType` Enumeration 142
Table 164: `SpindleInterlockClassType` Definition 143
Table 165: `ActiveStateDataType` Enumeration 143
Table 166: `PathModeClassType` Definition 143
Table 167: `PathModeDataType` Enumeration 144
Table 168: `PowerStateClassType` Definition 144
Table 169: `OnOffDataType` Enumeration 144
Table 170: `ProgramEditClassType` Definition 145
Table 171: `ProgramEditDataType` Enumeration 145
Table 172: `RotaryModeClassType` Definition 145
Table 173: `RotaryModeDataType` Enumeration 146
Table 174: `InterfaceStateClassType` Definition 146
Table 175: `InterfaceStatusDataType` Enumeration 146
Table 176: `MaterialFeedClassType` Definition 147
Table 177: `InterfaceStateDataType` Enumeration 147
Table 178: `MaterialChangeClassType` Definition 147

Table 179: `InterfaceStateDataType` Enumeration 148
Table 180: `MaterialRetractClassType` Definition 148
Table 181: `InterfaceStateDataType` Enumeration 148
Table 182: `MaterialLoadClassType` Definition 149
Table 183: `InterfaceStateDataType` Enumeration 149
Table 184: `MaterialUnloadClassType` Definition 149
Table 185: `InterfaceStateDataType` Enumeration 150
Table 186: `OpenDoorClassType` Definition 150
Table 187: `InterfaceStateDataType` Enumeration 150
Table 188: `CloseDoorClassType` Definition 151
Table 189: `InterfaceStateDataType` Enumeration 151
Table 190: `OpenChuckClassType` Definition 151
Table 191: `InterfaceStateDataType` Enumeration 152
Table 192: `CloseChuckClassType` Definition 152
Table 193: `InterfaceStateDataType` Enumeration 152
Table 194: `PartChangeClassType` Definition 153
Table 195: `InterfaceStateDataType` Enumeration 153
Table 196: `MTNumericEventClassType` Definition 154
Table 197: `AxisFeedrateOverrideClassType` Definition 154
Table 198: `BlockCountClassType` Definition 155
Table 199: `HardnessClassType` Definition 155
Table 200: `LineNumberClassType` Definition 156
Table 201: `PartCountClassType` Definition 156
Table 202: `RotaryVelocityOverrideClassType` Definition 156
Table 203: `MTStringEventClassType` Definition 157
Table 204: `BlockClassType` Definition 158
Table 205: `CoupledAxesClassType` Definition 158
Table 206: `LineLabelClassType` Definition 159
Table 207: `MaterialClassType` Definition 159
Table 208: `OperatorIdClassType` Definition 159
Table 209: `PalletIdClassType` Definition 160
Table 210: `PartIdClassType` Definition 160
Table 211: `PartNumberClassType` Definition 160
Table 212: `ProgramClassType` Definition 161
Table 213: `ProgramEditNameClassType` Definition 161
Table 214: `ProgramHeaderClassType` Definition 161
Table 215: `ProgramCommentClassType` Definition 162
Table 216: `SerialNumberClassType` Definition 162
Table 217: `ToolAssetIdClassType` Definition 162
Table 218: `ToolNumberClassType` Definition 163
Table 219: `ToolOffsetClassType` Definition 163
Table 220: `UserClassType` Definition 163
Table 221: `WireClassType` Definition 164
Table 222: `WorkholdingClassType` Definition 164
Table 223: `WorkOffsetClassType` Definition 164

Table 224: `MessageClassType` Definition . . . 165
Table 225: `AssetChangedClassType` Definition . . . 165
Table 226: `AssetRemovedClassType` Definition . . . 165
Table 227: `LineClassType` Definition . . . 166
Table 228: `ActuatorClassType` Definition . . . 166
Table 229: `CommunicationsClassType` Definition . . . 166
Table 230: `DataRangeClassType` Definition . . . 167
Table 231: `HardwareClassType` Definition . . . 167
Table 232: `LogicProgramClassType` Definition . . . 167
Table 233: `MotionProgramClassType` Definition . . . 168
Table 234: `SystemClassType` Definition . . . 168
Table 235: `MTDataItemSubClassType` Definition . . . 169
Table 236: `AbsoluteSubClassType` Definition . . . 172
Table 237: `ActualSubClassType` Definition . . . 172
Table 238: `ActionSubClassType` Definition . . . 172
Table 239: `AllSubClassType` Definition . . . 173
Table 240: `AlternatingSubClassType` Definition . . . 173
Table 241: `AScaleSubClassType` Definition . . . 173
Table 242: `AuxiliarySubClassType` Definition . . . 174
Table 243: `BadSubClassType` Definition . . . 174
Table 244: `BrinellSubClassType` Definition . . . 174
Table 245: `BScaleSubClassType` Definition . . . 175
Table 246: `CommandedSubClassType` Definition . . . 175
Table 247: `GoodSubClassType` Definition . . . 175
Table 248: `ControlSubClassType` Definition . . . 176
Table 249: `CScaleSubClassType` Definition . . . 176
Table 250: `DelaySubClassType` Definition . . . 176
Table 251: `DirectSubClassType` Definition . . . 177
Table 252: `DryRunSubClassType` Definition . . . 177
Table 253: `DScaleSubClassType` Definition . . . 177
Table 254: `FixtureSubClassType` Definition . . . 178
Table 255: `IncrementalSubClassType` Definition . . . 178
Table 256: `JobSubClassType` Definition . . . 178
Table 257: `KineticSubClassType` Definition . . . 179
Table 258: `LateralSubClassType` Definition . . . 179
Table 259: `LeebSubClassType` Definition . . . 179
Table 260: `LengthSubClassType` Definition . . . 180
Table 261: `LinearSubClassType` Definition . . . 180
Table 262: `LineSubClassType` Definition . . . 180
Table 263: `LoadedSubClassType` Definition . . . 181
Table 264: `MachineAxisLockSubClassType` Definition . . . 181
Table 265: `MaintenanceSubClassType` Definition . . . 181
Table 266: `ManualUnclampSubClassType` Definition . . . 182
Table 267: `MaximumSubClassType` Definition . . . 182
Table 268: `MinimumSubClassType` Definition . . . 182

Table 269: `MohsSubClassType` Definition . . . 183
Table 270: `MoleSubClassType` Definition . . . 183
Table 271: `MotionSubClassType` Definition . . . 183
Table 272: `NoScaleSubClassType` Definition . . . 184
Table 273: `OperatingSubClassType` Definition . . . 184
Table 274: `OperatorSubClassType` Definition . . . 184
Table 275: `OptionalStopSubClassType` Definition . . . 185
Table 276: `OverrideSubClassType` Definition . . . 185
Table 277: `PrimarySubClassType` Definition . . . 185
Table 278: `PoweredSubClassType` Definition . . . 186
Table 279: `ProbeSubClassType` Definition . . . 186
Table 280: `ProcessSubClassType` Definition . . . 186
Table 281: `ProgrammedSubClassType` Definition . . . 187
Table 282: `RadialSubClassType` Definition . . . 187
Table 283: `RapidSubClassType` Definition . . . 187
Table 284: `RelativeSubClassType` Definition . . . 188
Table 285: `RemainingSubClassType` Definition . . . 188
Table 286: `RequestSubClassType` Definition . . . 188
Table 287: `ResponseSubClassType` Definition . . . 189
Table 288: `RockwellSubClassType` Definition . . . 189
Table 289: `RotarySubClassType` Definition . . . 189
Table 290: `SetUpSubClassType` Definition . . . 190
Table 291: `ShoreSubClassType` Definition . . . 190
Table 292: `StandardSubClassType` Definition . . . 190
Table 293: `SwitchedSubClassType` Definition . . . 191
Table 294: `TargetSubClassType` Definition . . . 191
Table 295: `ToolChangeStopSubClassType` Definition . . . 191
Table 296: `ToolEdgeSubClassType` Definition . . . 192
Table 297: `ToolGroupSubClassType` Definition . . . 192
Table 298: `ToolSubClassType` Definition . . . 192
Table 299: `UasbleSubClassType` Definition . . . 193
Table 300: `VerticalSubClassType` Definition . . . 193
Table 301: `VolumeSubClassType` Definition . . . 193
Table 302: `VickersSubClassType` Definition . . . 194
Table 303: `WeightSubClassType` Definition . . . 194
Table 304: `WorkingSubClassType` Definition . . . 194
Table 305: `WorkpieceSubClassType` Definition . . . 195
Table 306: `«HasMTClassType»` Definition . . . 196
Table 307: `«HasMTComposition»` Definition . . . 196
Table 308: `«HasMTReference»` Definition . . . 197
Table 309: `«HasMTSource»` Definition . . . 197
Table 310: `«HasMTSubClassType»` Definition . . . 197
Table 311: `NamespaceMetadata` *Object* for this Specification . . . 199
Table 312: MTConnect *Server* Information Model . . . 200
Table 313: MTConnect *Client* Information Model . . . 200

Table 314: Namespaces used in a MTConnect Server 201
Table 315: Namespaces used used in this specification 201

OPC Foundation and MTConnect® Institute

AGREEMENT OF USE

All terms of use defined in documents provided by the OPC Foundation and the MTConnect Institute and referenced in this document are hereby incorporated and shall apply in their entirety into this document. Any conflict in terms between referenced documents and terms defined in this document shall default in priority to the terms defined in the original referenced documents.

COPYRIGHT RESTRICTIONS

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

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OPC or the MTConnect Institute specifications may require use of an invention covered by patent rights. OPC Foundation or the MTConnect Institute shall not be responsible for

identifying patents for which a license may be required by any OPC or the MTConnect Institute specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OPC or the MTConnect Institute specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

WARRANTY AND LIABILITY DISCLAIMERS

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

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

RESTRICTED RIGHTS LEGEND

This Specification is provided with Restricted Rights. Use, duplication or disclosure by the U.S. government is subject to restrictions as set forth in (a) this Agreement pursuant to DFARs 227.7202-3(a); (b) subparagraph (c)(1)(i) of the Rights in Technical Data and Computer Software clause at DFARs 252.227-7013; or (c) the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 subdivision (c)(1) and (2), as applicable. Contractor / manufacturer are the OPC Foundation, 16101 N. 82nd Street, Suite 3B, Scottsdale, AZ, 85260-1830 and MTConnect Institute, 7901 Jones Branch Dr., Suite 900, McLean, VA 22102-3316

COMPLIANCE

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

conformance with this specification.

TRADEMARKS

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

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

GENERAL PROVISIONS

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

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

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

1 Scope

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

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

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

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

The technologies provided from these two organizations include:

OPC Foundation

OPC is an interoperability standard for the secure and reliable exchange of data and information in the industrial automation space and in other industries. It is platform independent and ensures the seamless flow of information among devices from multiple vendors. The OPC Foundation is responsible for the development and maintenance of this standard. OPC UA is a platform independent service-oriented architecture that integrates all the functionality of the individual OPC Classic specifications into one extensible framework. This multi-layered approach accomplishes the original design specification goals of:

- Platform independence: from an embedded microcontroller to cloud-based infrastructure
- Secure: encryption, authentication, authorization and auditing

- Extensible: ability to add new features including transports without affecting existing applications
- Comprehensive information modelling capabilities: for defining any model from simple to complex

MTConnect Institute

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

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

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

2 OPC Unified Architecture for MTConnect Companion Specification Goals

This companion specification has been develop to provide the following:

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

3 Who Will Find Benefit from this Companion Specification?

This companion specification is written for those who:

- Have a clear understanding of both MTConnect and OPC UA.
- Wish to implement communication products that incorporate the combined functions of both standards.
- Support backend solutions with OPC UA Server and MTConnect Agent.
- Develop or want to develop client software applications with OPC UA and MTConnect.
- Have a reasonable level of programming expertise.

4 Normative References

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

4.1 OPC UA References

[UA Amend 1] *OPC UA Specification Amendment 1: Analog Types*. URL: https://opcfoundation.org/developer-tools/specifications-unified-architecture/errata-and-amendments/.

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

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

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

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

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

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

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

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

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

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

4.2 MTConnect References

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

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

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

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

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

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

4.3 Other References

[Fie00] Roy Thomas Fielding. “Architectural Styles and the Design of Network-based Software Architectures”. PhD thesis. 2000. ISBN: 0599871180. DOI: 10.1.1.91.2433.

5 Terms, Definitions and Conventions

5.1 Overview

The basic concepts of OPC UA and MTConnect are pre-requisites for understanding and interpreting the content provided in this companion specification. Additionally, the terms and definitions given in [UA Part 01], [UA Part 02], [UA Part 03], [UA Part 05], [UA Part 07], [UA Part 10], and [MTConnect Part 1.0], (see section 4), as well as the following, apply to this document.

5.2 Conventions

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

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

5.3 Terms and Acronnyms

5.3.1 Conventions for Node descriptions

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

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

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

- If the ***TargetNode*** is a component of the ***Node*** being defined in the table the ***Attributes*** of the composed Node are defined in the same row of the table.
- The **`DataType`** is only specified for Variables; "[<number>]" indicates a single-dimensional array, for multi-dimensional arrays the expression is repeated for each dimension (e.g. [2][3] for a two-dimensional array). For all arrays the ***ArrayDimensions*** is set as identified by **`<number>`** values. If no **`<number>`** is set, the corresponding dimension is set to 0, indicating an unknown size. If no number is provided at all the ***ArrayDimensions*** can be omitted. If no brackets are provided, it identifies a scalar **`DataType`** and the **`ValueRank`** is set to the corresponding value

(see [UA Part 03]). In addition, ***ArrayDimensions*** is set to **`null`** or is omitted. If it can be **`Any`** or **`ScalarOrOneDimension`**, the value is put into **`"<value>"`**, so either **`"Any"`** or **`"ScalarOrOneDimension"`** and the **`ValueRank`** is set to the corresponding value (see [UA Part 03]) and the ***ArrayDimensions*** is set to **`null`** or is omitted. Examples are given in Table 1.

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

Table 1: Examples of DataTypes

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

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

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

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

Table 2: Type Definition Table

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

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

5.3.2 NodeIds and BrowseNames

5.3.2.1 NodeIds

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

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

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

Note that this specification not only defines concrete ***Nodes***, but also requires that some Nodes shall be generated, for example one for each Session running on the Server. The

NodeIds of those ***Nodes*** are Server-specific, including the namespace. But the ***NamespaceIndex*** of those ***Nodes*** cannot be the ***NamespaceIndex*** used for the Nodes defined in this specification, because they are not defined by this specification but generated by the Server.

5.3.2.2 BrowseNames

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

5.3.3 Common Attributes

5.3.3.1 General

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

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

Table 3: Common Node Attributes

Attribute	Value
DisplayName	The DisplayName is a LocalizedText. Each server shall provide the DisplayName identical to the BrowseName of the Node for the LocaleId "en". Whether the server provides translated names for other LocaleIds is server-specific.
Description	Optionally a server-specific description is provided.
NodeClass	Shall reflect the NodeClass of the Node.
NodeId	The NodeId is described by BrowseNames.
WriteMask	Optionally the WriteMask Attribute can be provided. If the WriteMask Attribute is provided, it shall set all non-server-specific Attributes to not writable. For example, the Description Attribute may be set to writable since a Server may provide a server-specific description for the Node. The NodeId shall not be writable, because it is defined for each Node in this specification.
UserWriteMask	Optionally the UserWriteMask Attribute can be provided. The same rules as for the WriteMask Attribute apply.
RolePermissions	Optionally server-specific role permissions can be provided.
UserRolePermissions	Optionally the role permissions of the current Session can be provided. The value is server-specifc and depend on the RolePermissions Attribute (if provided) and the current Session.
AccessRestrictions	Optionally server-specific access restrictions can be provided.

5.3.3.2 Objects

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

Table 4: Common Object Attributes

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

5.3.3.3 Variables

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

Table 5: Common Variable Attributes

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

5.3.3.4 VariableTypes

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

Table 6: Common VariableTypes Attributes

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

5.3.3.5 Methods

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

Table 7: Common Method Attributes

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

6 Introduction to MTConnect and OPC UA

6.1 MTConnect

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

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

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

the information provided by these data models between an MTConnect *Agent* (server) and a client software application.

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

6.1.1 Data Dictionary

The Data Dictionary defines a consistent set of terms that are used to describe information and data gathered from shop floor operations. When various pieces of equipment publish information using this common Data Dictionary, that data is easier to understand and can be used directly for further analysis without requiring additional manipulation to get the data into a common format. By utilizing the Data Dictionary, equipment can now publish data that this "self-describing" - meaning the data not only provides values, but also provides essential meaning for the data; including units, tag names, scaling information, and any other information that may be needed for a software application to fully understand both the meaning of the data and the relevance of that data to the manufacturing process. The users of this data no longer have to define this information each place the data is used.

6.1.2 Semantic Data Models

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

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

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

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

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

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

6.1.3 Fundamentals of MTConnect

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

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

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

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

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

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

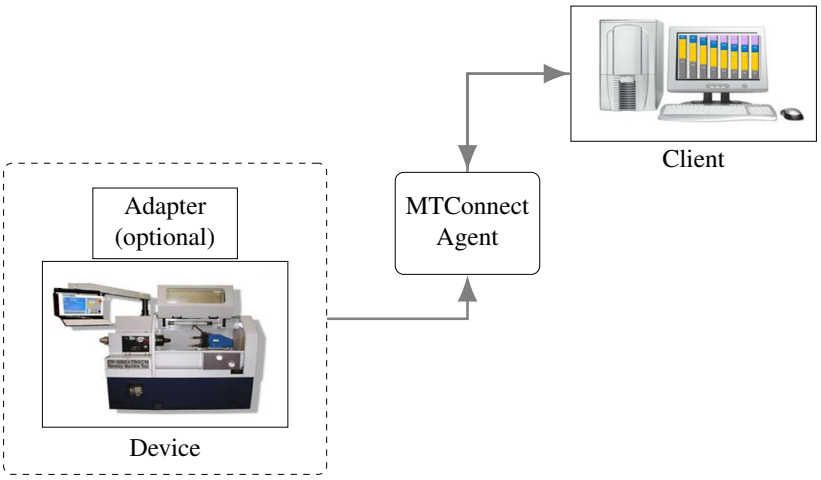

Figure 1: MTConnect Architecture Overview

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

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

6.2 Introduction to OPC Unified Architecture

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

- A state of art security model (see [UA Part 02]).
- A fault tolerant communication protocol.
- An information modeling framework that allows application developers to represent their data in a way that makes sense to them.

OPC UA has a broad scope which delivers for economies of scale for application developers. This means that a larger number of high quality applications at a reasonable cost are available. When combined with powerful semantic models such as MTConnect, OPC UA makes it easier for end users to access data via generic commercial applications.

The OPC UA model is scalable from small devices to enterprise resource planning (ERP) systems. OPC UA devices process information locally and then provide that data in a consistent format to any application requesting data - ERP, manufacturing execution system (MES), Production Management System (PMS), Maintenance Systems, Human Machine Interface (HMI), Smartphone or a standard Browser, for examples. For a more complete overview see [UA Part 01].

6.2.1 Basics of OPC UA

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

As an Extensible Standard, OPC UA provides a set of services (see [UA Part 04]) and a basic information model framework. This framework provides an easy manner for creating and exposing vendor defined information in a standard way. More importantly all OPC UA Clients are expected to be able to discover and use vendor defined information. This means OPC UA users can benefit from the economies of scale that come with generic visualization and historian applications. This specification is an example of an OPC UA Information Model designed to meet the needs of developers and users.

OPC UA Clients can be any consumer of data from another device on the network to browser base thin clients and ERP systems. The full scope of OPC UA applications are shown in Figure 2.

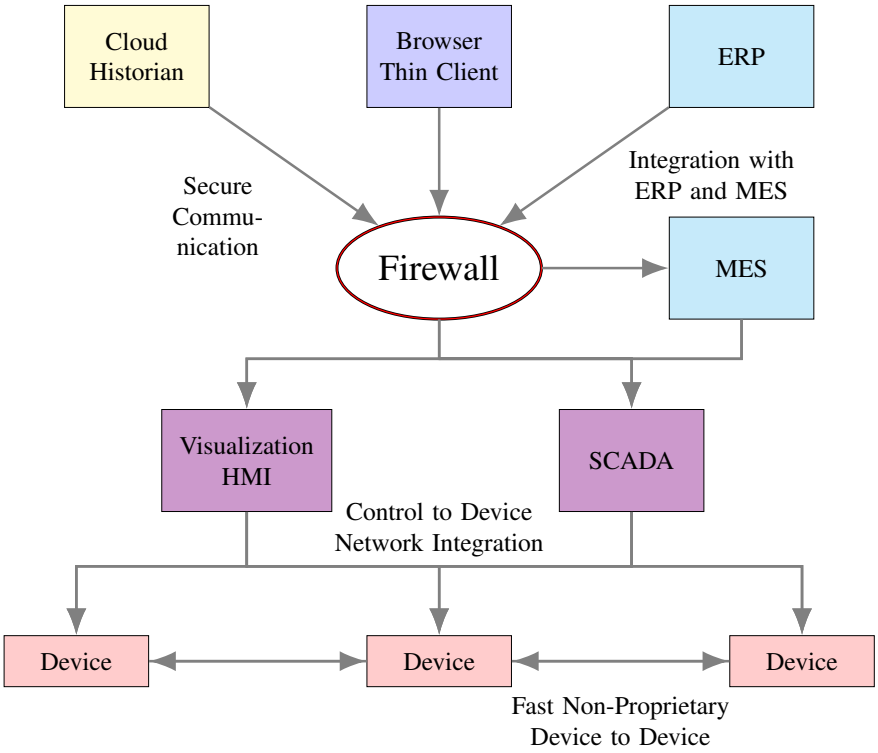

Figure 2: The Scope of OPC UA within an Enterprise

OPC UA provides a robust and reliable communication infrastructure having mechanisms for handling lost messages, failover, heartbeat, etc. With its binary encoded data, it offers a high-performing data exchange solution. Security is built into OPC UA as security requirements become more and more important especially since environments are connected to the office network or the internet and attackers are starting to focus on automation systems.

6.2.2 Information Modeling in OPC UA

6.2.2.1 Concepts

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

3.

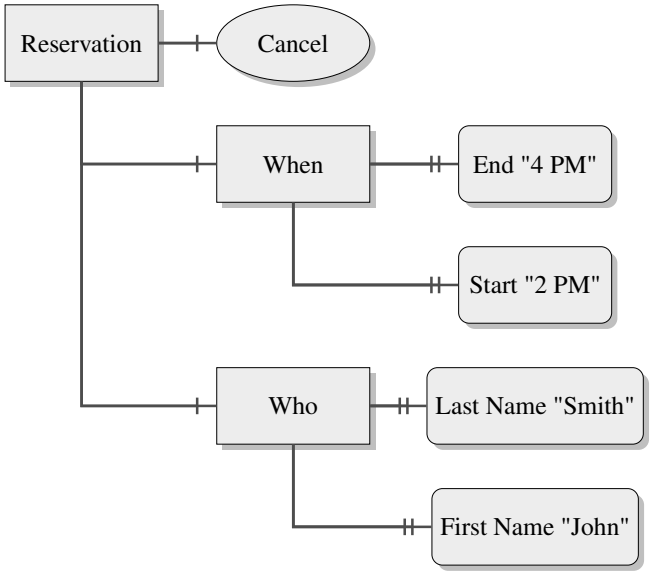

Figure 3: A Basic Object in an OPC UA Address Space

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

The Type Nodes are templates that define all of the children that can be present in an Instance of the Type. In the example in Figure 4 the `PersonType ObjectType` defines two children: First Name and Last Name. All instances of `PersonType` are expected to have the same children with the same **`BrowseNames`**. Within a Type the **`BrowseNames`** uniquely identify the child. This means Client applications can be designed to search for children based on the **`BrowseNames`** from the Type instead of `NodeIds`. This eliminates the need for manual reconfiguration of systems if a Client uses Types that multiple devices implement.

OPC UA also supports the concept of sub typing. This allows a modeler to take an existing Type and extend it. There are rules regarding sub typing defined in [UA Part 03], but in general they allow the additions to a given type or the restriction of a **`DataType`** to a more specific data type. For example the modeler may decide that the existing `ObjectType` in some cases needs an additional variable. The modeler can create a subtype of the `ObjectType` and add the variable. A client that is expecting the parent type can treat the new `ObjectType` as if it was of the parent `ObjectType` and just ignore the additional variable. A client that understands the new subtype may display or otherwise process the additional variable. With regard to **`DataTypes`**, if a variable is defined to have a numeric value, a sub type could restrict the Value to a float.

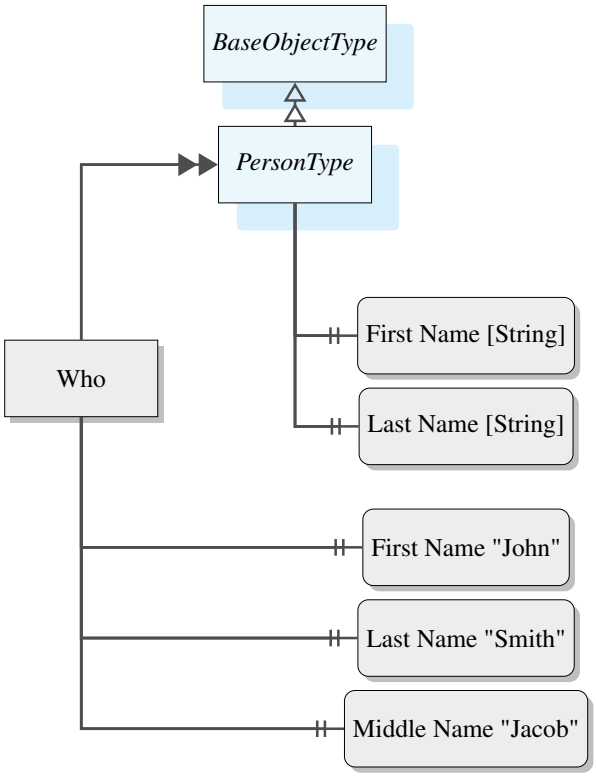

Figure 4: The Relationship between Type Definitions and Instances

References allow Nodes to be connected together in ways that describe their relationships. All References have a `ReferenceType` that specifies the semantics of the relationship. References can be hierarchical or non-hierarchical. Hierarchical references are used to create the structure of Objects and Variables. Non-hierarchical are used to create arbitrary associations. Applications can define their own `ReferenceType` by creating `Subtypes` of the existing `ReferenceType`. `Subtypes` inherit the semantics of the parent but may add additional restrictions. Figure 5 and Figure 6 depict several references connecting different Objects.

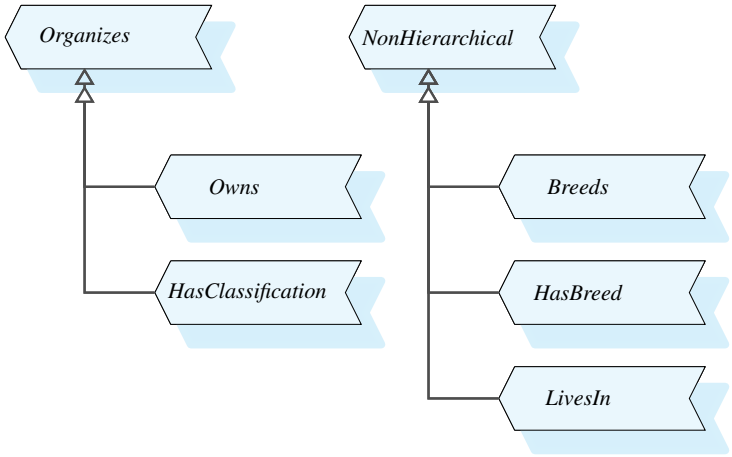

Figure 5: Reference Types from other Reference Types

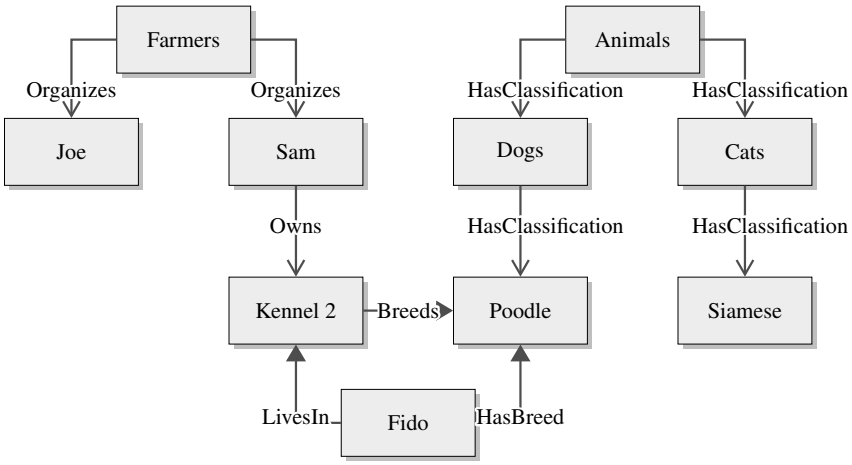

Figure 6: References showing hierarchies and/or relationships

The figures above use a notation that was developed for the OPC UA specification. The notation is summarized in Figure 7 and Figure 8 . UML representations can also be used; however, the OPC UA notation is less ambiguous because there is a direct mapping from the elements in the figures to ***Node*** in the `AddressSpace` of an OPC UA Server.

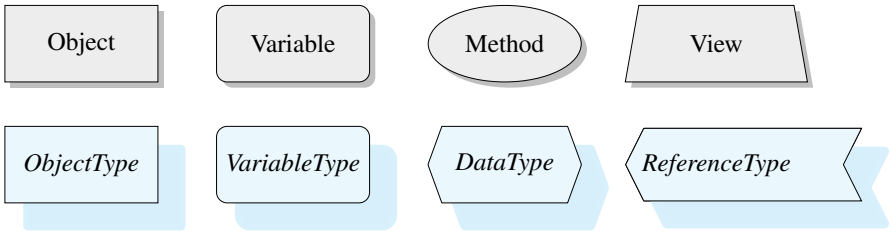

Figure 7: The OPC UA Information Model Notation for Types and Instances

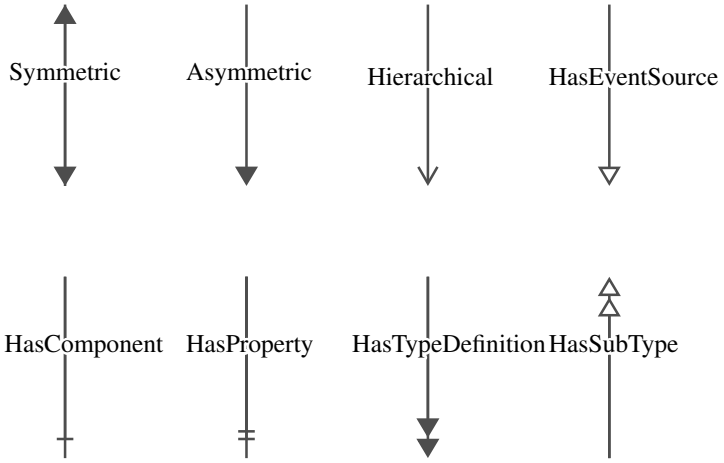

Figure 8: The OPC UA Standard References

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

6.2.2.2 Namespaces

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

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

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

6.2.2.3 Companion Specifications

An OPC UA companion specification for an industry specific vertical market describes an Information Model by defining ObjectTypes, VariableTypes, DataTypes and ReferenceTypes (see section 5.2) that represent the concepts used in the vertical market, and potentially also well-defined Objects as entry points into the AddressSpace.

7 Use Cases

MTConnect is concerned with getting semantic data or information from manufacturing systems so it can be analyzed and communicated to other pieces of equipment in the ecosystem. The goal of MTConnect is to provide the data in a way that can enable other systems to create value for the manufacturing industry.

The use cases examine the architecture from the perspective of multiple personae and how the MTConnect domain model enables technology that addresses their requirements. There are numerous use cases for MTConnect; many that do not exist because the ecosystem is not mature enough to support those use cases. The design of the semantic model envisioned the need for extensibility and new manufacturing processes that were not widely available when MTConnect began. The model in OPC UA is also extensible and will be able to track the evolution of MTConnect. As well, as OPC UA advances, MTConnect will revise to make use of the latest capabilities like publish/subscribe.

What follows is a non-exhaustive list of use cases illustrating the benefits of the MTConnect domain-specific semantic information model when integrated with the OPC UA communication and information modeling framework to increase the community of users by leveraging the achievements of both standards bodies.

7.1 Machine Tool Manufacturer with Existing MTConnect Implementation

Figure 9, illustrates the use case for a manufacturer of a piece of equipment (device) that needs to connect to other systems. OPC UA provides the communication platform, and the MTConnect semantics provide meaning and structure to device data. Figure 9 shows several clients developed for different purposes that can access information produced by the device via OPC UA.

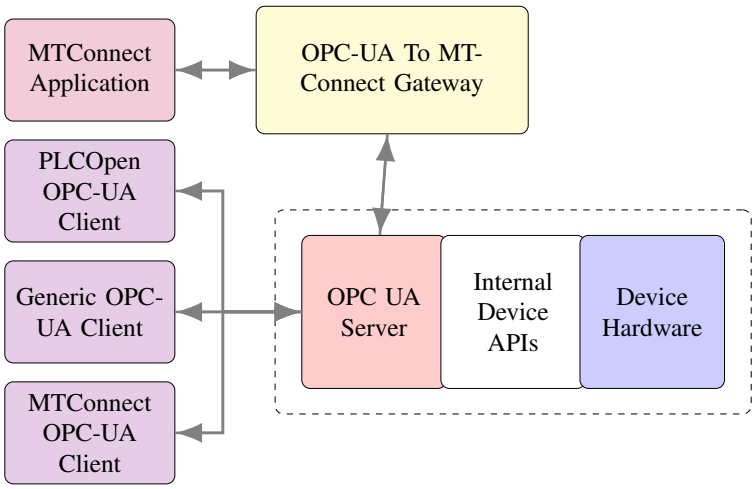

Figure 9: The Device Manufacturer Use Case

The MTConnect or OPC UA interface may reside directly on the machine or on a separate device that communicates with the machine. The location for the interface is up to the implementer.

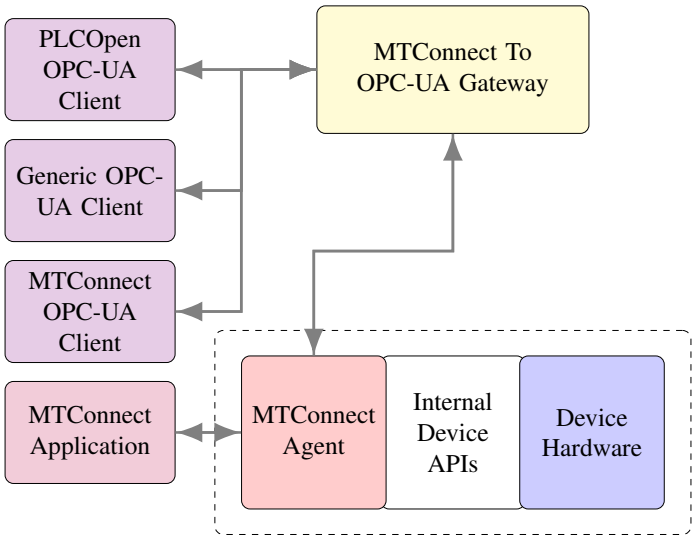

Figure 10: Device Manufacturer with Native MTConnect Agent

The device manufacturer may also have a native MTConnect device and make use of an MTConnect to OPC UA gateway to provide information to OPC UA clients (see Figure 10 and Figure 11); this companion specification allows for information flow between clients and servers that support either MTConnect or OPC UA.

The benefit of providing options allows the largest number of application access to the data and enables the application developers to require the least amount of effort to ingest and analyze the data. The goal of the device manufacturer is to make their equipment capable of participating in the ecosystem of manufacturing technology that will improve the value and effectiveness of their products.

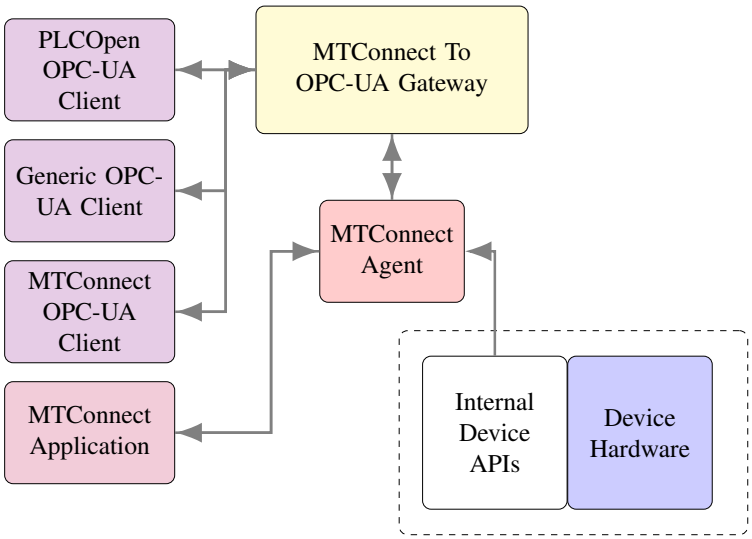

Figure 11: Device Manufacturer with Separate MTConnect Agent

7.2 Software Vendor

An Independent Software Vendor (ISV) has the principle concern of connecting to as many pieces of equipment as possible as quickly as possible. There are three aspects of interoperability that address this concern, protocol, syntactic and semantic. The first concern requires that the application can communicate to the piece of equipment using a protocol that is well understood by both the sender and receiver and widely used across the industry.

Protocols like HTTP, *MODBUS*, and Message Queuing Telemetry Transport (MQTT) provide a level of protocol connectivity and communication with well-understood request and response formats that open and widely available. These protocols do not provide an information model; they provide access to data as an undefined set of bytes that need to be interpreted by the application.

The next level of interoperability is syntactic. Frameworks like OLE for Process Control (OPC), Unified Modeling Language (UML), and Ontology Web Language (OWL) provide

a foundation with the ability to express an *ontology*, but does not provide the *ontology* itself. OPC provides a model that has been designed for the industrial domain but is generic across many verticle applications.

The last level is the semantic interoperability where they provide the meaning and structure of the data, the *ontology*, and allow for the information to be understood and analyzed. From an application perspective, the ability to rapidly implement solutions requires that data be well-understood with definitions for all the information so that the solution does not need to be modified for every piece of equipment.

Figure 12 illustrates the use case for an ISV supplying products to industrial equipment users. A typical ISV offering includes gateway(s) that convert information between MTConnect and OPC UA and may also provide additional features required for MTConnect implementations; e.g. enhanced security features. The OPC Unified Architecture for MTConnect Companion Specification allows the ISV to extend the MTConnect-OPC UA information model with application specific constructs. These can be easily accessed via any standard OPC UA client product and will function in parallel to existing features provided by MTConnect. Figure 12 shows an ISV product that consumes data from MTConnect and OPC UA enabled devices and then makes it available via MTConnect and OPC UA.

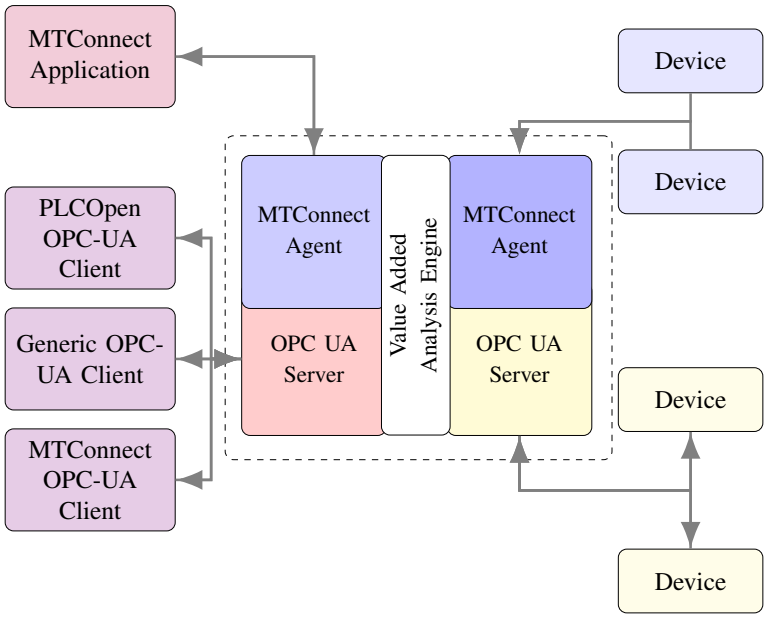

Figure 12: The Independent Software Vendor (ISV) Use Case

7.3 Data Scientist

Cloud-based analytics platforms, such as Microsoft Azure® Cloud provide factory connectivity solutions based on OPC UA for data ingest and analysis. Data Scientists, like ISVs, need semantic data that where the acquisition and categorization of the incoming

data do not dominate the project. The primary focus of a data scientist is to find interesting insights; to do so requires the information has normalized units and vocabulary so deeper correlations can be uncovered.

With a common information model and representations of other aspects of the manufacturing process that utilizes the OPC information model, such as PLC Open and ISA-95, the information can be combined with additional context to provide a better understanding of the impact on the manufacturing systems and provide more profound insights to the users.

7.4 Industrial Systems Integrator

[MTConnect Part 5.0] provides a data-centric approach to equipment integration using an observation based architecture where the devices do not instruct each other, but they observe the requests of different pieces of equipment and respond by providing the necessary services. The model uses MTConnect ability to publish changes to special event *DataItems* that provide a request/response framework to allow the equipment to organize activities. This is the section of the MTConnect standard referred to as *interfaces*.

MTConnect can benefit from OPC UA's discovery mechanisms to incorporate devices as they enter the interactive ecosystem. UA will also increase the available equipment that can participate in the MTConnect Interfaces model by supplying a standard for advanced industrial automation without the need for additional Programmable Logic Controllers (PLCs) and expensive centralized control systems.

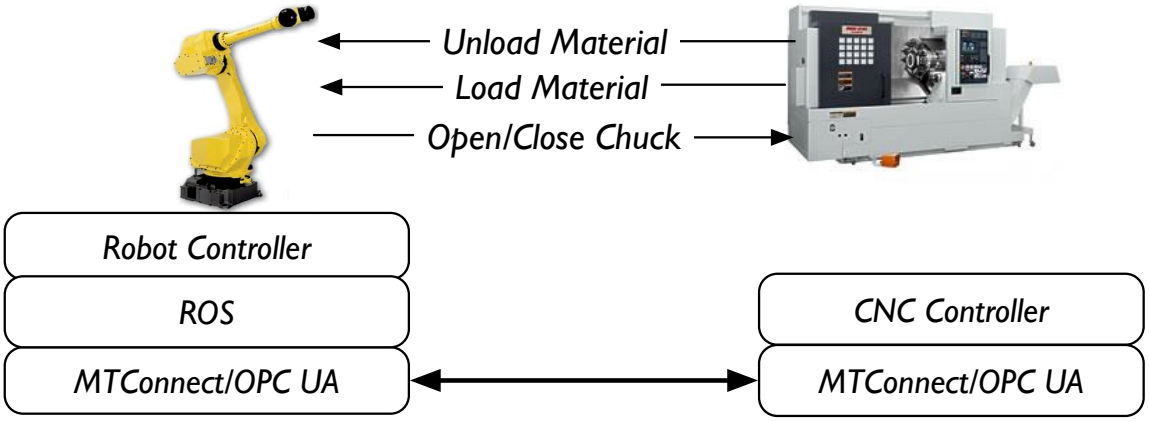

Figure 13: MTConnect OPC UA and ROS Device Integration

Figure 13 shows how two devices can interact without any additional components. The combination of the two standards allows for more extensive semantic interoperability between equipment and applications, but also semantic interoperability between manufacturing equipment.

8 Mapping the MTConnect Information Model to OPC UA

This section describes a UML representation of MTConnect semantic data models for mapping MTConnect into OPC Unified Architecture (UA). More detailed information is provided in Section 9 for each data type.

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

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

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

8.1 MTConnect UML Representation of OPC

This section provides instructions to translate from the OPC UA diagram representation given in section 6.2 to the UML described throughout section 8. Figure 14 is a partial illustration of the MTConnect *Components* abstraction in OPC UA.

Figure 15 represents the same model in UML. The companion specification uses the following conventions:

- OPC UA Types are represented as UML Classes
- OPC UA Objects are represented as "**`BrowseName`**: ***Type***" underlined as in the following example: **SimpleCnc: MTDeviceType** where **SimpleCnc** is the name of the device and **MTDeviceType** is related with a **`HasTypeDefinition`** relationship.
- Stereotypes are used to denote properties of types, relationships, and members of

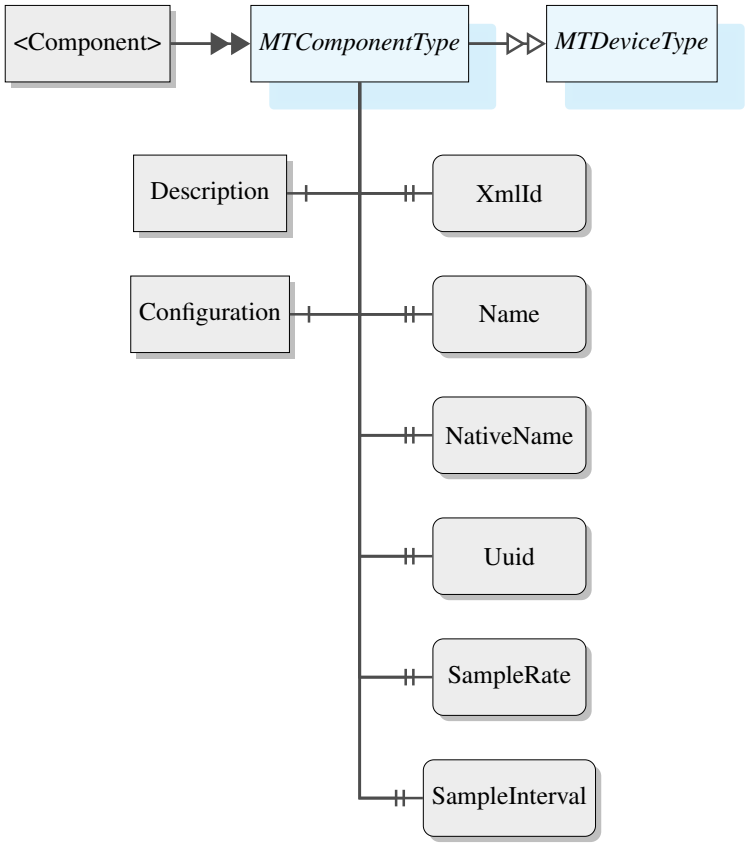

Figure 14: MTConnect MTComponentType in OPC UA

types. A stereotype is denoted by angle brackets («..») and provides additional information about the entity. For example, each reference from one ***ObjectType*** to another has a stereotype denoting the reference type.

- OPC UA ***Properties*** are denoted as UML Attributes. UA ***Attributes*** are also represented as UML Attributes with the stereotype of «`attribute`» to differentiate from UA ***Properties***.
- OPC UA **`HasComponent`** ***References*** are represented as UML Unidirectional associations. The reference type is given as the association's `Stereotype`, for example «`HasComponent`».
- **`HasSubtype`** relationships are given as an UML Generalization association.
- Combinining of templates and classes can be done using a UML *Realization* with the stereotype *«Mixes In»* to use common properties and relationships in a collection of otherwise unrelated types.

The *UMLAssociations* are used to represent OPC UA ***References***. UA ***Properties*** represented using the UA **`HasProperty`** relationships are given as UML ***Attributes*** (not to be confused with the OPC UA ***Attribute***. UML allows for additional information as follows: the multiplicity of a property, the data type of the property, are specified.

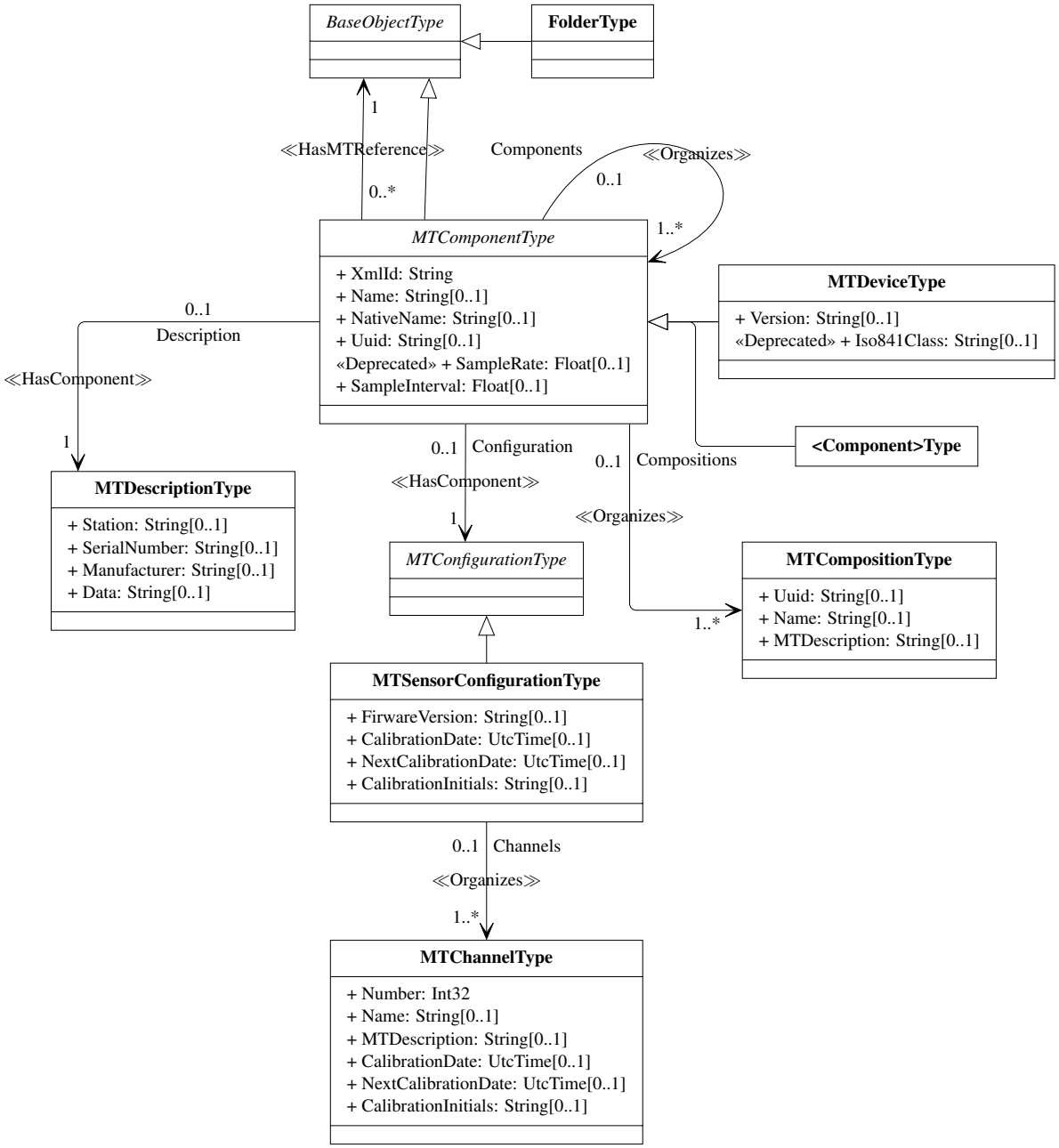

Figure 15: MTConnect MTComponentType in UML

When traversing an association between two object types, the name of the source of the association is the **BrowseName** of the object, and the destination is the type of object that is instantiated. If the name is not given, it represents a dynamic relationship where the **BrowseName** is determined during the creation of the object model. An example is the association of the MTConnect *DataItems*. The rules for creating the **BrowseName** are givin below in Section 8.3.1.

A reference that has a stereotype of **Organizes**, shown by the *Components* recursive relationship in the MTComponentType, implies an intermediate object of type **FolderType** with the **BrowseName** of the relationship point on the source side. The far side is not constrained within the **FolderType** but given in the documentation as the expected contents of the folder.

OPC UA represents both class and instance diagrams using the same set of primitives. UML provides separate class and object models using two separate sets of diagram and representations, one for classification and the other for instances of those classes. This section separates the object instantiation from the classification and uses UML object diagrams to represent example instances of classes.

8.1.1 MTConnect UML Model

The models will use a UML representation that represents the members as UML *Slots*. There is no distinction between an attribute and a property; consult the type model for the proper associations.

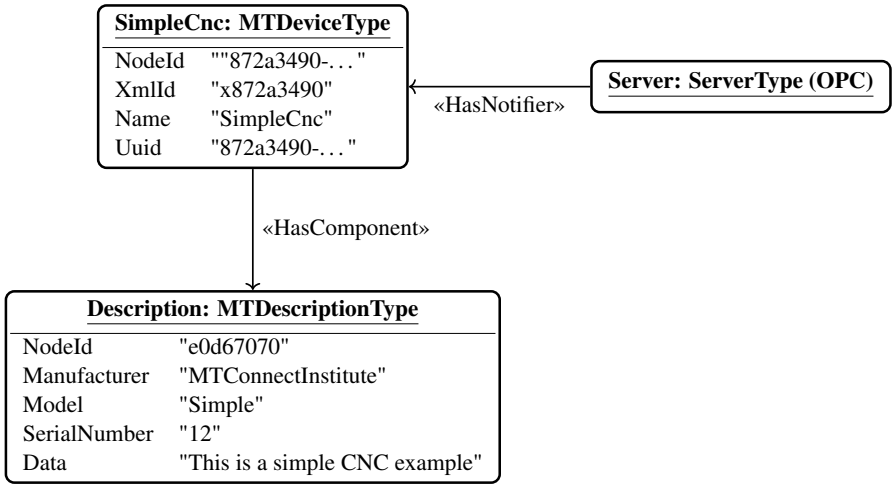

Figure 16: MTConnect Example Object

Figure 16 represents an MTConnect *Device* with the associated values. The UA **BrowseName** is given as the object name above, and the UA Type is given after the **:** in the header. Each of the UA ***Attributes*** and ***Properties*** are provided below. Relationships are shown as edges that have the Reference Type, such as **HasComponent**. The properties contained

within will be created using the **HasProperty** reference but composed as internal members for brevity.

For example, the with type MTDeviceType has **BrowseName** of SimpleCnc (the name of the device). There are patterns for creating all the **BrowseName** for each MTConnect UA Type. The rules are presented and in the example to provide the normative patterns for each type in the information model.

Common elements, like *Components* or *Compositions*, that provide structure are mapped to **FolderType** objects have common glsBrowseNames. These will be prefixed with the context of the parent component. For example, Controller.Components will refer to an ***Object*** of type **FolderType** with the **BrowseName** *Components*. The prefix before the period (.) will be ignored for the **BrowseName**.

8.2 MTConnect Information Model

The MTConnect information model has the following abstractions:

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

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

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

Assets are complex information models that provide a point in time consistent set of information about the use of a physical or logical entity in the manufacturing process. These models, for example, may represent a cutting tool, a program, or a process. The *Assets*

737 will be covered in a subsequent companion specification. The only assets currently in
738 the MTConnect standard are `CuttingTool` and `CuttingToolArchetype`. Refer
739 to MTConnect Part 4.0 [MTConnect Part 4.0] and MTConnect Part 4.1 [MTConnect Part
740 4.1].

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

745 8.3 Mapping The Model

746 The MTConnect information model is represented in the UA address space as set of ***Ob-***
747 ***ject*** and ***Variable Types***. Most have pre-existing type definitions whereas others will be
748 created dynamically following rules in the following section. The rules provide a way for
749 a developer of a server to extend the nodeset to include the types present in a given MT-
750 ConnectDevices XML document as presented from an MTConnect *Agent* in response to a
751 *probe request*.

752 8.3.1 Mapping Rules and Conventions

753 MTConnect has conventions regarding the format of elements, attributes, and values of
754 attributes or CDATA the represent controlled vocabularies or enumerations.

755 When the rules refer to *Pascal Case*, the entity must have the following structure: The
756 first letter of each word is capitalized and the remaining letters are in lowercase. All
757 space is removed between letters. For example, *controller mode override* becomes `Con-`
758 `trollerModeOverride`. The one exception is `PH` that remains `PH`.

759 The document refers to *Lower Camel Case* when the first word is lowercase and the re-
760 maining words are capitalized and all spaces between words are removed. For example,
761 *controller mode override* is written as `controllerModeOverride`.

762 The third form used in MTConnect is uppercase with underscores separating characters.
763 For example, *controller mode override* is `CONTROLLER_MODE_OVERRIDE`.

764 The rules are as follows:

765 - XML Elements are in *Pascal Case*.

766 - XML Attributes are in *Lower Camel Case*

- *Controlled Vocabulary*, a fixed or enumerated set of values, for attribtues or the CDATA of elements are all caps with an underscore ("_") separating words.

When the value of an attribute, such as the data item type as given in Listing 1:

Listing 1: `DataItem type` Attribtue Conversion

```
1   <DataItem category="EVENT" id="a8ce34a00" name="mode_ovr"
        type="CONTROLLER_MODE_OVERRIDE"/>
2   <DataItem category="EVENT" id="b8ce34a00" name="eob" type
        ="END_OF_BAR"/>
```

The type names in UA will require the upper case with underscore name to be converted to *Pascal Case*. XML Attributes are converted from *Lower Camel Case* to *Pascal Case* as well when mapping to UA Property **`BrowseNames`**.

OPC UA conventions use names in *Pascal Case*, so all MTConnect entities will be represented using *Pascal Case*. When the MTConnect entities are converted into UA types, they are converted to *Pascal Case* and the word `Type` is appended.

The values of the attributes and enumerations are represented in MTConnect in capitals with underscores. OPC UA has the same convention, so these types will remain unchanged when converted to OPC UA. The only caveat is that enumerated values when represented in UA **`DataVariable`**s are numeric. All enumerations in MTConnect are sorted alphabetically and assigned monotonically increasing numeric values. For example, the enumeration for the `Execution` MTConnect data item is in table 8.

Table 8: `ExecutionDataType` Enumeration

Name	Index
ACTIVE	0
FEED_HOLD	1
INTERRUPTED	2
OPTIONAL_STOP	3
READY	4
PROGRAM_COMPLETED	5
PROGRAM_STOPPED	6
STOPPED	7

Extensions to the MTConnect standard with enumerations will represent the data as strings since the values are not included in the address space. An extension nodeset can be created by a vendor if the `MTControlledVocabEventType` is desired. The **`EnumValues`** needs to reference the **`EnumValues`** ***Attribute*** of the UA **`MultiStateValueDiscreteType`** and the associated indexes with the translation to the text values for mapping.

MTConnect base types have been prefixed with the letters "MT" to avoid confusion with OPC UA vocabulary. The following component types and relationships have been annotated:

- Configuration → MTConfigurationType
- Component → MTComponentType
- Composition → MTCompositionType
- Description → MTDescriptionType
- Device → MTDeviceType
- Message → MTMessageType
- SensorConfiguration → MTSensorConfigurationType

The following types representing base MTConnect types are also prefixed:

- HasMTClassType
- HasMTSubClassType
- MTAssetEventType
- MTConditionType
- MTControlledVocabEventType
- MTNumericEventType
- MTSampleType
- MTThreeSpaceSampleType

The data item class types and enumeration data types do not have "MT" prepended.

8.3.2 Component and Composition `BrowseName` and *Type* Rules

The rules for determining the **`BrowseName`** for the MTConnect *Component* are as follows:

1. If the *Component* is *Device*, the **`BrowseName`** is the `name` attribute of the `Device` and the UA ***Type Definition*** is `MTDeviceType`.
2. The browse name is the `QName` of the XML Element of the *Component*.

3. The ***HasTypeDefinition*** references a type that is created by appending the word ***Type*** to the `QName` of the *Component* element. For example, a `Controller` will have a **`BrowseName`** of `Controller` and a **`HasTypeDefinition`** referencing `ControllerType`.

4. The ***ObjectType*** is only defined once for all *Components* of that type. There is no `MT` prepended to the front of the ***ObjectType BrowseName***.

5. If there are two *Components* with the same **`BrowseNames`** or if the *Component* are `Linear` or `Rotary`, they must have the `name` attribute appended in square brackets (`[]`) to the name as specified in item 2.

6. For example, if there are two `Path` *Components* with names `name="P1"` and `name="P2"`, the **`BrowseNames`** will be as follows: `Path[P1]` and `Path[P2]` where each **`HaveTypeDefinition`** of `PathType`.

The rules for *Composition* elements are as follows:

1. *Composition* elements in the MTConnect model will be represented instantiated with the **`BrowseName`** of the *Composition* `type` converted to *Pascal Case*.

2. If two *Composition* elements of the same *Component* have the same `type`, they must have the `name` attribute appended in square brackets (`[]`) to the name as specified in item 1.

3. For example, a component has two `type="TANK"`s with the names `name="top"` and `name="bottom"`, the **`BrowseNames`** will be `Tank[top]` and `Tank[bottom]` respectively.

4. The type is placed is the `MTTypeName` ***Property***.

8.3.3 `DataItem` `HasTypeDefinition` and `BrowseName` Conventions

The MTConnect *DataItem* represents multiple OPC UA **`DataVariableType`**s as well as UA **`ConditionType`**s as described in parts [UA Part 08] and [UA Part 09] respectively. The mapping rules are as follows:

1. Determine the *DataItem* ***Type Definition*** for the **`HasTypeDefinition`** ***Reference***:

   - Evaluate *DataItem* attributes as follows:

     (a) If the `category` is `CONDITION`, the ***Type Definition*** is `MTConditionType`.

(b) If the `category` is `SAMPLE`; apply the following rules:

i. If the `type` is `PATH_POSITION` then the ***Type Definition*** is `MTThreeSpaceSampleType`.

ii. Otherwise, the UA type is `MTSampleType`.

(c) If the `category` is `EVENT`; apply the following rules:

i. If the `type` is `ASSET_CHANGED` or `ASSET_REMOVED`; the UA ***Type Definition*** is `MTAssetEventType`

ii. Convert the `type` attribute to *Pascal Case* and append `ClassType`. This name will be referred to as *`@classType`*.

iii. Find the MTConnect UA *`@classType`* that corresponds to the value computed in the previous step. The types can be found in Sections `ControlledVocabDataItemTypes`, `NumericEventDataItemTypes`, and `StringEventDataItemTypes`.

iv. If the *`@classType`* matches one of the `StringEventDataItemTypes`; the MTConnect UA ***Type Definition*** is `MTStringEventType`.

v. If the *`@classType`* matches one of the `NumericEventDataItemTypes`; the MTConnect UA ***Type Definition*** is `MTNumericEventType`.

vi. If the *`@classType`* matches one of the `ControlledVocabDataItemTypes`, the MTConnect UA ***Type Definition*** is `MTControlledVocabEventType`. The **`EnumValues`** will be assigned as follows:

Each of the ***ObjectTypes*** matching the *`@classType`* for `ControlledVocabDataItemTypes` will have a ***Property*** **`EnumValues`**. The **`EnumValues`** will reference the **`Enumeration`** with the values of the *Controlled Vocabulary*.

vii. Otherwise, the UA ***Type Definition*** is `MTStringEventType`. This will apply to all extended types.

2. Assign the `HasMTClassType` and `HasMTSubClassType` ***References***

(a) Convert the *DataItem* `type` attribute to *Pascal Case* and append **`ClassType`**. Assign the target of the `HasMTClassType` to the ***ObjectType***.

(b) Put the text name of the class in the property `MTTypeName`.

(c) Convert the *DataItem* `subType` attribute to *Pascal Case* and append `SubClassType`. Assign the target of the `HasMTSubClassType` to the ***ObjectType***

(d) Put the text name of the class in the property `MTSubTypeName`.

3. Format the **`BrowseName`** for the `Sample` and `Event` *DataItems*

   (a) If the *DataItem* attribute `compositionId` is present, do the following:

      i. Resolve the `compositionId` to the *Composition* XML element with that `id`.

      ii. Set the variable *`@composition`* to the *Pascal Case* of the *Composition* attribute `type`.

   Otherwise, set the variable *`@composition`* to "".

   (b) If the `subType` is present, set the variable *`@subType`* to the *Pascal Case* of the `subType`. Otherwise set it the variable *`@subType`* to "".

   (c) Set the variable *`@type`* to the *Pascal Case* of the *DataItem* attribute `type`.

   (d) The **`BrowseName`** is the concatination of the *`@composition`* + *`@subType`* + *`@type`*.

   (e) If the `representation` attribute is given and it has a value other than the default `VALUE`, the **`BrowseName`** will be have the *Pascal Case* of the `representation` appended to the **`BrowseName`** as specified in 3d . For example, a `AMPERAGE` with `representation="TIME_SERIES"` will have a **`BrowseName`** of `AmperageTimeSeries`.

   (f) If the `statistic` is given, the *Pascal Case* of the `statistic` attribute will be prepended to the **`BrowseName`** as specified in 3e. For example, if the `statistic="AVERAGE"` for a *DataItem* of `AMPERAGE` and a `representation` of `TIME_SERIES`, the **`BrowseName`** will be `AverageAmperageTimeSeries`.

   (g) If two **`BrowseNames`** are identical for the same *Component*, the `name` must be used to differentiate between the *DataItems* by placing the `name` in square brackets (`[]`). For example, if there are two `TEMPERATURE` *DataItems* with the attributes `name="front"` and `name="back"` the *DataItems* will have the **`BrowseNames`** `Temperature[front]` and `Temperature[back]` respectively.

4. Format the **`BrowseName`** for the ***Condition*** *DataItems*

   (a) Use the name derived in the previous step for **`Events`** and `Samples` as specified in item 3f and append `Condition`.

   (b) Apply rule specified in item 3g if two conditions have identical **`BrowseNames`** to the result of item 4a.

5. The `type` is placed is the `MTTypeName` ***Property***.

The examples below will provide further guidance on implementing these rules.

8.3.4 Mapping `MTDataItem units` to `EngineeringUnits`

OPC UA defines the **`EngineeringUnits DataType`** in [UA Part 08] as having the fields show in Table 9.

Table 9: `EngineeringUnits` DataType structure

Name	Type	Description
namespaceUri	String	Identifies the organization (company, standards organization) that defines the EUInformation
unitId	Int32	Identifier for programmatic evaluation. -1 is used if a unitId is not available.
displayName	LocalizedText	The displayName of the engineering unit is typically the abbreviation of the engineering unit, for example "h" for hour or "m/s" for meter per second.
description	LocalizedText	Contains the full name of the engineering unit such as "hour" or "meter per second".

In Table 10, the mapping between the MTConnect units and the OPC UA EngineeringUnits structure has been specified. There is one unit type, `COUNT` that is unitless and therefor not mapped. The `COUNT` is only used with **`Events`** and therefor is not an **`AnalogUnitType`**, and is instantiated as an `MTNumericEventType`.

The **`namespaceUri`** will be set to `http://www.opcfoundation.org/UA/units/un/cefact` as specified in [UA Part 08] and the values from Table 10 must be used..

The `MILLIMETER_3D` unit is special since it will be represented by the `MTThreeSpaceSampleType`. The individual values of the `ThreeSpacePositionDataType` will be given in *millimeters* and a **`displayName`** of $mm(\mathbb{R}^3)$; the **`EngineeringUnits`** will be provided for consistency. The **`EURange`** will not work since it cannot represent three space volumetric constraints and the **`EURange`** ***Property*** will not be created.

Table 10: `EngineeringUnits` DataType structure

MTConnect Unit	UNECE Code	UnitId	Display Name	Description
AMPERE	AMP	4279632	A	Amps
CELSIUS	CEL	4408652	$^{\circ}Celsuis$	Degrees Celsius
COUNT				A counted event Cannot be used for EUInformation
DECIBEL	2N	12878	dB	Sound Level
DEGREE	DD	17476	$^{\circ}$	degree [unit of angle]
DEGREE/SECOND	E96	4536630	$^{\circ}/s$	Angular degrees per second
DEGREE/SECOND^2	M45	5059637	$^{\circ}/s^2$	Angular acceleration in degrees per second squared
HERTZ	HTZ	4740186	Hz	Frequency measured in cycles per second
JOULE	JOU	4869973	J	A measurement of energy.
KILOGRAM	KGM	4933453	kg	kilogram
LITER	LTR	5002322	l	Litre
LITER/SECOND	G51	4666673	l/s	Litre per second
MICRO_RADIAN	B97	4340023	μrad	microradian - Measurement of Tilt
MILLIMETER	MMT	5066068	mm	millimetre
MILLIMETER/SECOND	C16	4403510	mm/s	millimetre per second
MILLIMETER/SECOND^2	M41	5059633	mm/s^2	Acceleration in millimeters per second squared
MILLIMETER_3D	MMT	5066068	$mm(\mathbb{R}^3)$	A point in space identified by X, Y, and Z coordinates.
NEWTON	NEW	5129559	N	Force in Newtons
NEWTON_METER	NU	20053	$N \cdot m$	Torque, a unit for force times distance.
OHM	OHM	5195853	Ω	Measure of Electrical Resistance
PASCAL	PAL	5259596	Pa	Pressure in Newtons per square meter
PASCAL_SECOND	C65	4404789	$Pa \cdot s$	Measurement of Viscosity
PERCENT	P1	20529	$\%$ or pct	Percent
PH	Q30	5321520	pH	pH (potential of Hydrogen) - A measure of the acidity or alkalinity of a solution
REVOLUTION/MINUTE	RPM	5394509	r/min	revolutions per minute
SECOND	SEC	5457219	s	second [unit of time]

Table 11: `EngineeringUnits` DataType structure (Continued)

MTConnect Unit	UNECE Code	UnitId	Display Name	Description
SIEMENS/METER	D10	4469040	S/m	siemens per metre - A measurement of Electrical Conductivity
VOLT	VLT	5655636	V	volt
VOLT_AMPERE	D46	4469814	VA	volt - ampere
VOLT_AMPERE_REACTIVE		-1	VAR	Volt-Ampere Reactive (VAR)
WATT	WTT	5723220	W	watt
WATT_SECOND	J55	4863285	$W \cdot s$	Measurement of electrical energy, equal to one Joule

8.3.5 Mapping Example

The following is an example of mapping the MTConnect Device Meta-Model to the OPC UA MTConnect model. The example will not be a realistic machine tool configuration, but will demonstrate all the possible combinations of components, compositions, and data items and how to construct the information model using the various rules given above. The first section covers the meta model presented in [MTConnect Part 2.0] of the MTConnect standard. For more information on any individual entity in the model, please refer to the MTConnect documentation.

8.3.5.1 `MTConnectDevices` Root Element

The first lines shown in Listing 2 of the MTConnect XML representation are the root element `MTConnectDevices` and the `Header` element that is used for the MTConnect protocol. The only root node concern is if there are additional name-spaces declared in the root `MTConnectDevices` element. The other area of note is the `version` attribute of the `Header` element. The version indicates the most current version of the MTConnect standard currently being provided by this *Agent*. The remaining attributes are relevant during the discussion of streaming data in the following section.

Listing 2: Device Header

```
1 <MTConnectDevices xmlns="urn:mtconnect.org:MTConnectDevices:1.4"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:
    schemaLocation="urn:mtconnect.org:MTConnectDevices:1.4 schema
    /MTConnectDevices_1.4.xsd">
2   <Header version="1.4.0" creationTime="2018-10-28T12:33:12Z"
      instanceId="12345" sender="localhost" bufferSize="1024"
      assetBufferSize="1024" assetCount="0"/>
```

```
3  <Devices>
```

The MTConnect *Agent* is capable of supporting multiple *Devices*; a `Device` element is the only allowed child element of the `Devices` element. The *Device* element is the top level *Component* of the MTConnect *Component* hierarchy. The *Device* is a sub-type of the *Component* and inherits all the structure of an MTConnect *Component*. The `Device` has three required attributes, an `id`, a `uuid` and a `name` attribute. Only the `id` is mandatory in all other components.

8.3.5.2 `Device` Element

Listing 3: `Device` Element Mapping

```
4     <Device id="x872a3490" uuid="872a3490-bd2d-0136-3eb0-0
          c85909298d9" name="SimpleCnc">
5       <Description manufacturer="MTConnectInstitute" model="
            Simple" serialNumber="12">
6                                 This is a simple CNC example
7                         </Description>
```

The `Description` element provides some characteristics of the device, namely the `manufacturer`, `serialNumber` and `model`. These are all optional as is the descriptive text that is contained in the `Description` element. The text contained in the `CDATA` of the `Description` element is mapped to the `Data` property of the `MTDescriptionType` instance.

The `Device` is mapped to an `MTDeviceType` object in the MTConnect namespace as shown in Figure 17.

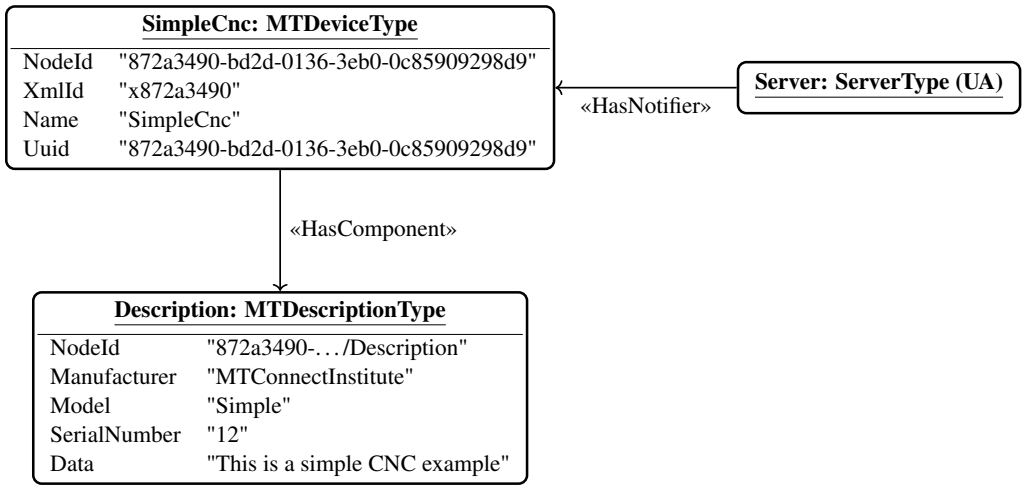

Figure 17: MTConnect Device Object

Every device must have a **HasNotifier** relationship with the ***Server*** UA Object as the source to enable conditions and event notification. The **HasNotifier** relationships are flowed down through the component hierarchy where the ***Objects*** each have a **HasNotifier** relationship with the parent. The MTConnect *DataItems* will have require a **HasEventSource** relationship with the parent *Component* when the *DataItem* is associated with a `Condition` where the `Source` element of the `Condition` references the *DataItem* `id`.

The treatment of the component relationships will be covered in the *DataItem* examples.

8.3.5.3 `Device` *DataItems*

Listing 4: Device Data Items

```
8       <DataItems>
9         <DataItem id="d5b078a0" name="avail" type="AVAILABILITY"
      category="EVENT"/>
10        <DataItem id="e4a300e0" type="ASSET_CHANGED" category="
              EVENT"/>
11        <DataItem id="f2df7550" type="ASSET_REMOVED" category="
              EVENT"/>
12      </DataItems>
```

The device requires three data items as of MTConnect version 1.2. The `AVAILABILITY` data item indicates if data is available from the device and the `ASSET_CHANGED` and `ASSET_REMOVED` data items represent the last asset inserted or updated and removed respectively. The Component adds Data items with a ***HasComponent*** relationship, and the BrowseName is constructed using the `type`, `subType`, and related *Composition* from `compositionId` if given. A complete example is given later in this section.

Figure 18 demonstrates the relationship of the *DataItems* with the parent `MTComponentType` using a **HasComponent** relationship. The browse name is constructed to uniquely identify the DataItem and also indicate its semantic meaning within the context of the MTConnect `MTComponentType`, in this case, an `MTDeviceType` since the `MTDeviceType` is a sub-type of the `MTComponentType`. These relationships are created dynamically during the instantiation of the model and are not part of the underlying MTConnect Nodeset.

Every MTConnect *DataItem* has two references that provide the semantic meaning in the UA address space. This model is very similar to the **ConditionClassId** relationship for the OPC UA ***ConditionType***. These **ClassTypes** are inherited from the same parent so they can be used for both the `HasMTClassType` for all *DataItems* as well as the **ConditionClassId** in ***Conditions*** since MTConnect `Conditions` are also descended from the ***ConditionType*** hierarchy.

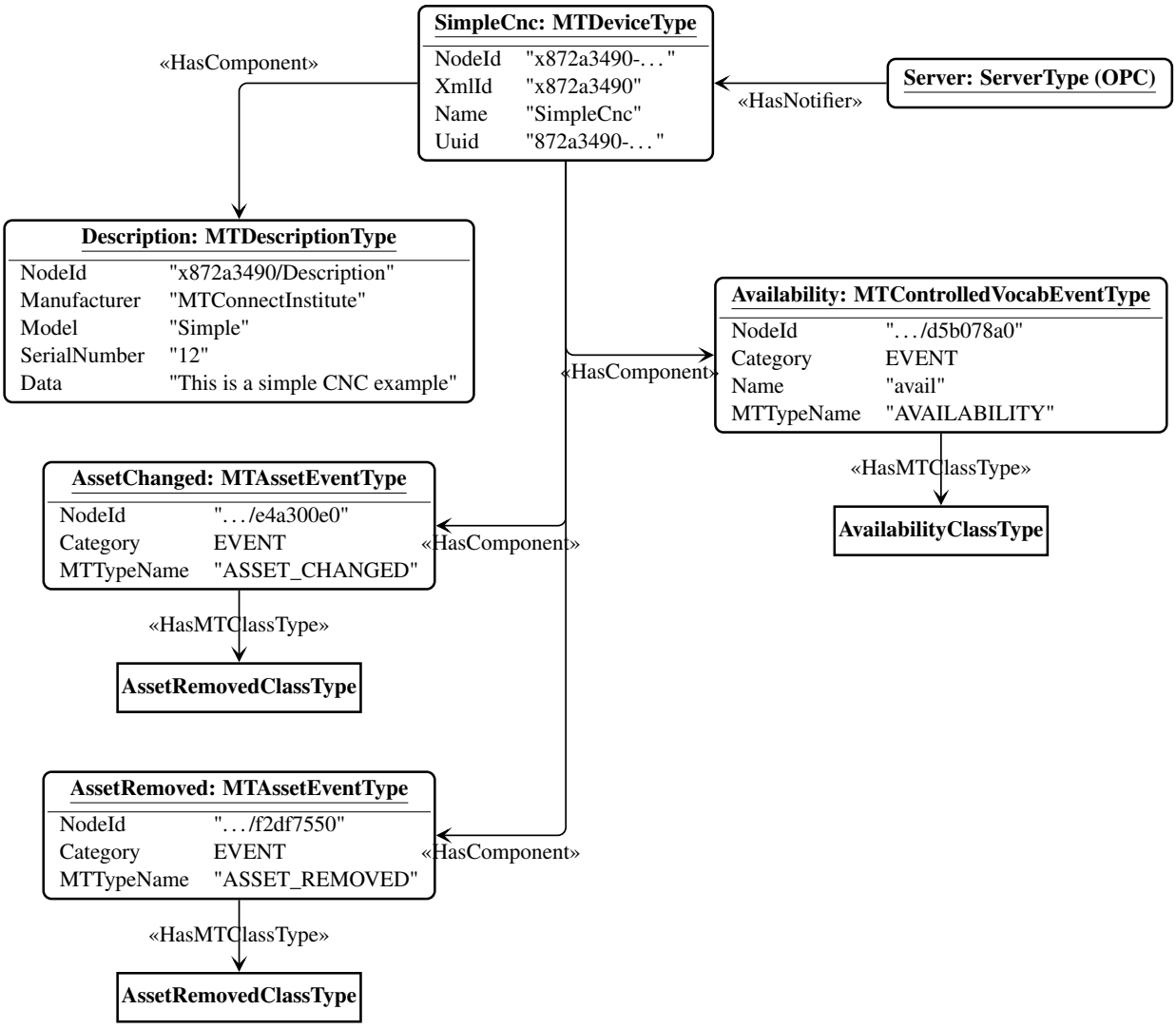

Figure 18: MTConnect Device Object with Data Items

In Figure 19, the `HasMTClassType` reference indicates the semantic meaning of the data item. For enumerated or controlled vocabulary data items, they are represented using a sub-type of the UA **`MultiStateValueDiscreteType`**, the `MTControlledVocabEventType`, where the **`EnumValues`** attribute is related to the NodeId of the associated enumeration data type, in this case the `AvailabilityDataType`. The `AvailabilityClassType` also has **`HasComponent`** relationship with the `AvailabilityDataType` **`Enumeration`**.

MTConnect `subType` will be mapped in the same manner as the type using the `HasMTSubClassType` relationship. This will be covered subsequent examples.

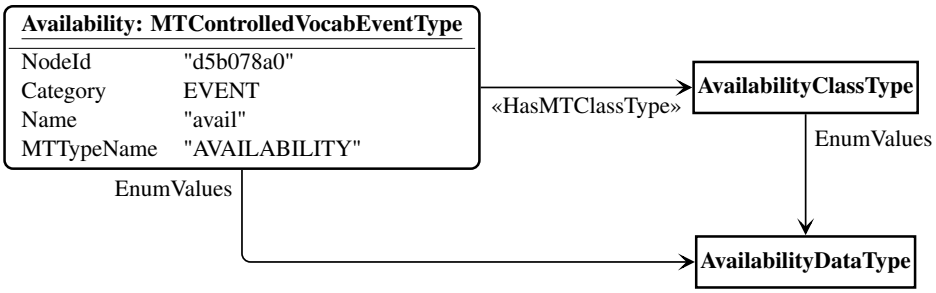

Figure 19: MTConnect Data Item References

8.3.5.4 `Axes` and `Linear` Components

Listing 5: Components and Conditions

```
13      <Components>
14        <Axes id="a62a1050">
15          <Components>
16            <Linear id="e373fec0" name="X1" nativeName="X">
17              <DataItems>
18                <DataItem id="dcbc0570" name="Xpos" type="
    POSITION" subType="ACTUAL" category="SAMPLE" units="
    MILLIMETER"/>
19                <DataItem id="f646f730" type="LOAD" name="Xload"
                      category="SAMPLE" units="PERCENT"/>
20                <DataItem id="e086dd60" type="POSITION" category=
                      "CONDITION" name="Xtravel"/>
21              </DataItems>
22            </Linear>
```

The *Components* element in Listing 5 is represented as a UA **`FolderType`**. Within the folder, the MTConnect components are semantically identified by the element's `QName` with a type with the suffix `Type` appended, in this case the **`BrowseName`** `Axes` has a **`HasTypeDefinition`** relaton of `AxesType`.

Figure 20 represents the UA Object model based on Listing 5. The `Linear` X `Axis`, mapped from `Linear` element in Listing 5, has a **`BrowseName`** composed of the `QName` of the element, and the `name` attribute appended and enclosed in square brackets `[X]`, giving the browse name of `Linear[X]` and a component type of `LinearType` using a **`HasTypeDefinition`** ***Reference***.

The *DataItems* for `ActualPosition` and `Load` are both of `category SAMPLE`; they contain numeric values are always mapped to the `MTSampleType` which is derived from the UA **`AnalogItemType`** defined in [UA Part 08]. The **`BrowseName`** for the *DataItems* are composed of the related *Composition*, `subType`, and `type` of the *DataItem* represented in *Pascal Case*. See Section 8.3.3 for details.

The ***EngineeringUnits*** of the ***AnalogUnitType*** will be mapped to the ***EngineeringUnits***

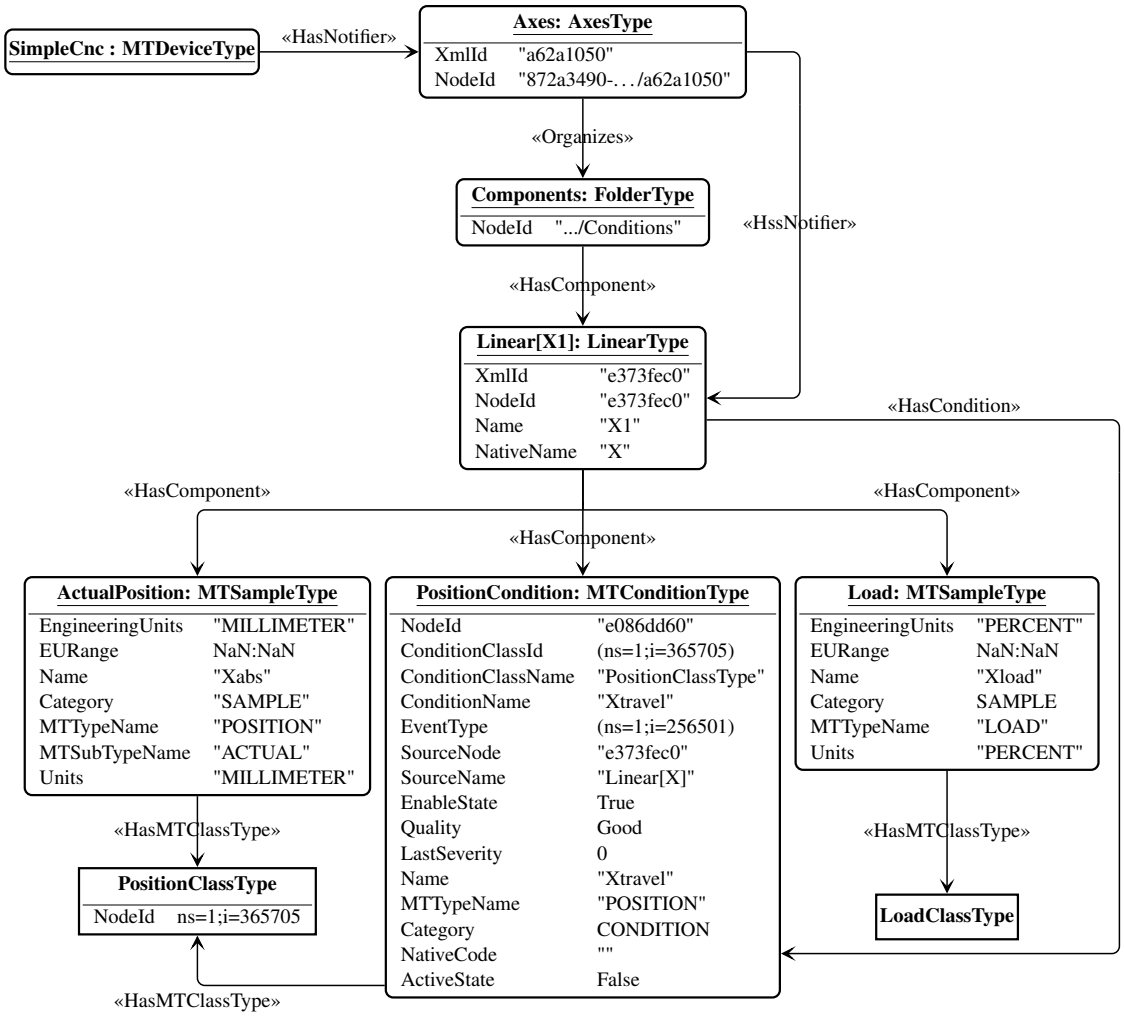

Figure 20: Linear X Axis Example

Property of the `MTSampleType`. If the value is constrained, the ***EURange*** will be created, otherwise it will not be specified. The remaining attributes will be represented as properties of the `MTSampleType` converting the *Lower Camel Case* to *Pascal Case* by changing the first character to upper-case. The *EVENT* category and complex cases will be specified in the following examples.

MTConnect *DataItems* of `category CONDITION` are special types since they represent the state of an alarm or warning associated with a *Component* of the *Device*. For more information on the *conditions* see [MTConnect Part 2.0] and [MTConnect Part 3.0]. The mapping of the `CONDITION category` of *DataItems* is to the `MTConditionType`.

The `MTConditionType` has a naming rule similar to the *DataItem* referenced above. The same rule applies with `Condition` appended. In this case the condition is of `type POSITION` so the **`BrowseName`** is `PositionCondition`. This is to differentiate the `Conditions` from other *DataItems*.

The MTConditionType is decendent of the **BaseEventType** described in [UA Part 05] and [UA Part 09], so it has two functions in the UA model. It represents the overall state of the UA ***Condition*** and also it represents each event as the various MTConnect Condition changes are reported as represented in the MTConnectStreams described in [MTConnect Part 3.0].

The **HasNotifier** reference must be created between the SimpleCnc:MTDeviceType object and the Axes object to allow for the events to flow down to the conditions for each of the components. The relationship of the condition in Figure 20 illustrates the relationships between the Component and the related condition in the notification hierarchy.

The Linear[x] component uses a **HasNotifier** reference sourced from the parent **Axes** ***Object***. The condition, having a MTClassType and a **ConditionClassId** of PositionClassType, is connected to the Linear[X] axis with a **HasCondition** relationship to indicate the source of the events for the ***ConditionType***.

8.3.5.5 Rotary Axis and Condition sources

Listing 6: Rotary C Axis

```
23          <Rotary id="zf476090" name="C" nativeName="S">
24            <DataItems>
25              <DataItem id="bbe3f010" type="ROTARY_MODE"
    category="EVENT">
26                <Constraints>
27                  <Value>SPINDLE</Value>
28                </Constraints>
29              </DataItem>
30              <DataItem id="ac6b69c0" type="ROTARY_VELOCITY"
                  subType="PROGRAMMED" category="SAMPLE" units=
                  "REVOLUTION/MINUTE" name="Sspeed_prg"/>
31              <DataItem id="vee9c2d0" type="ROTARY_VELOCITY"
                  subType="ACTUAL" category="SAMPLE" units="
                  REVOLUTION/MINUTE" name="Sspeed_act">
32              <Constraints>
33                <Minimum>0</Minimum>
34                <Maximum>7000</Maximum>
35              </Constraints>
36              </DataItem>
37              <DataItem id="r1841b70" type="LOAD" category="
                  SAMPLE" units="PERCENT" name="Sload"/>
38              <DataItem id="taa7a0f0" type="AMPERAGE" category=
                  "SAMPLE" units="AMPERE" compositionId="
                  b7792870" />
39              <DataItem id="afb596b0" type="AMPERAGE" category=
                  "CONDITION" compositionId="b7792870" name="
                  Soverload">
40              <Source dataItemId="taa7a0f0"/>
41              </DataItem>
```

```
42                </DataItems>
43                <Compositions>
44                  <Composition id="b7792870" type="MOTOR"/>
45                </Compositions>
46              </Rotary>
```

The data item on Line 25 of Listing 6 with `type` of `ROTARY_MODE` is illustrated in Figure 21. MTConnect allows for a `Rotary` axis to provide the function of the axis, Spindle, Contour, or Index. In Many cases the Rotary Mode is a singular values, as in this case when it can only work as a spindle. This is expressed by creating a **`HasComponent`** reference to a `MTConstraintType` ***Object*** and specifying the **`Values`** ***Property*** with the singular value of `["SPINDLE"]` indicating it can only function as a spindle.

In MTConnect, the value of this data item will never be `UNAVAILABLE` since it is bound to a single value, it will always be present and have that value regardless of the connectivity state to the data source. Since the value is static and never changes, the **`Quality`** will be **`Uncertain_SubstituteValue`** if it is never provided. In the case of constraints on a controlled vocabulary, the **`Values`** must always be one of the related **`EnumValues`**. See [MTConnect Part 2.0] for details on the constraints and the relationship to the *DataItems*.

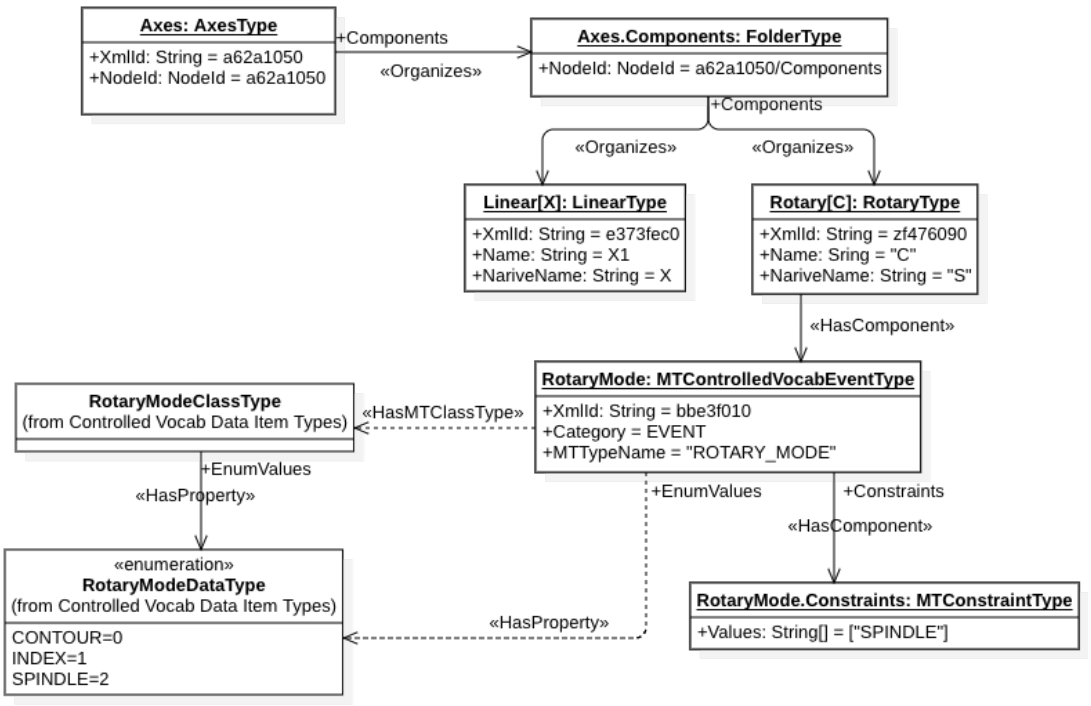

Figure 21: Rotary[C] Axis RotaryMode DataItem

Figure 22 corresponds to Lines 30-36 of Figure 22–the two rotary velocity data items differentiated by their `subTypes` of `ACTUAL` and `PROGRAMMED`. In this case they share many of the same property values. The `HasMTSubClassType` references the `Actu-`

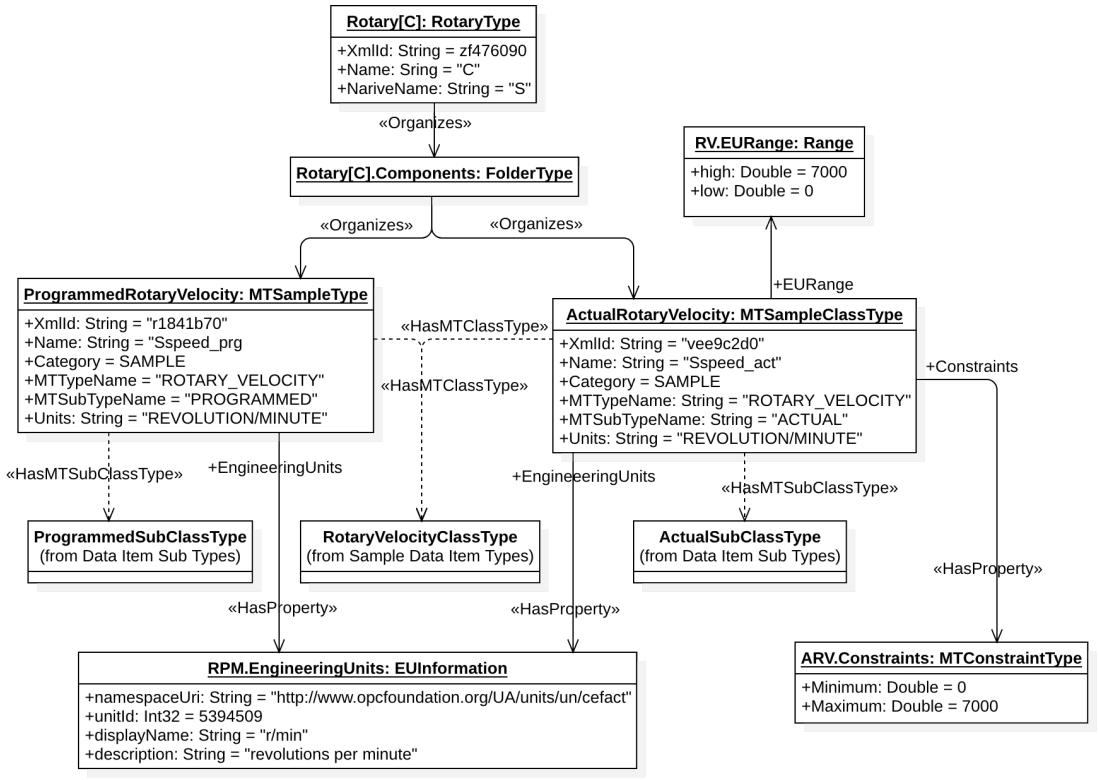

Figure 22: Rotary[C] Axis RotaryVelocity DataItem

alSubClassType and the ProgrammedSubClassType respectively.

In the case of the ActualRotaryVelocity object, it has an additional constraint that specifies that the maximum spindle speed is 7000 RPM. This is given with the MTConstraintType.

UA 1.04 Ammedment 1 [UA Amend 1] for **AnalogItem** ***Types*** provides the ***Type*** **AnalogUnitType** where the **EURange** is **Optional**. The MTSampleType will inherit from **AnalogUnitType** instead of **AnalogItemType** as this matches the MTConnect unit requirements.

If the ***Constraints*** are given with both *Maximum* and Minimum values, then the **EURange** will be created with **high** set to Maximum and **low** set to the Minimum, otherwise if either are not given, the ***EURange*** will not be created. The ActualRotaryVelocity references an **EURange** where the ***high*** and **low** are the same as the Maximum and Minimum of the MTConstraintType respectively.

Figure 23 represents the Load *DataItem* for Listing 6 Line 37 as an example of a MTSampleType with a units="PERCENT" and mapped to the ***EngineeringUnits*** for **Percent**. There is no *Constraints* so the EURange is not provided as specified in [UA Part 08].

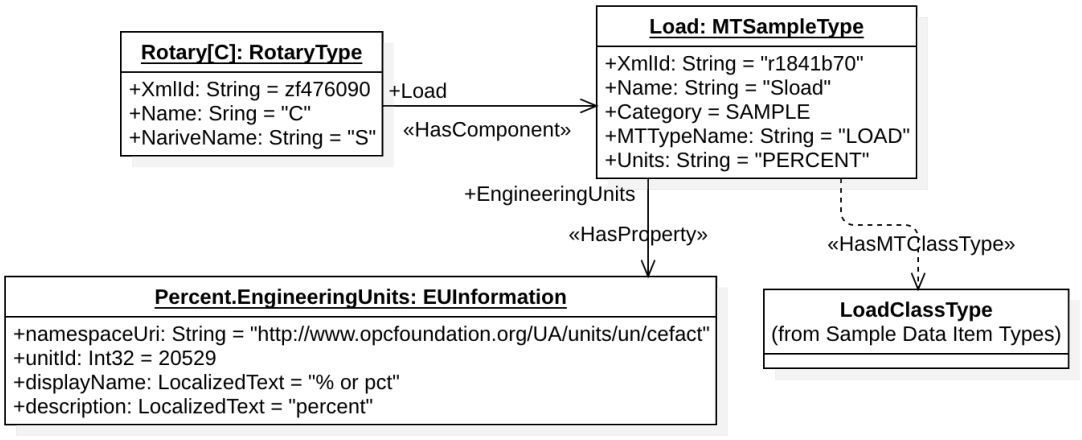

Figure 23: Rotary[C] Axis Load DataItem

Figure 24 illustrates the `Amperage` *DataItem* corresponding to the Listing 6 Lines 38-42. Unlike the previous example of a MTConnect `Condition` relating to the MTConnect *Component*, the `Source` element under the *DataItem* references the *DataItem* with `type AMPERAGE`. When the `Source` refers to a *DataItem* that is part of the same MTConnect *Component*, the UA **`Variable`** is given a ***Reference*** **`HasEventSource`** from the `RotaryType` (`Rotary[C]`) and a **`HasCondition`** ***Reference*** is created between the ***Variable*** to the `MTConditionType`.

As stated above, the **`HasNotifier`** ***References*** must be established between all *Components* back to the `MTDeviceType` that has a reference from the ***Server***. The `MotorAmperage` and `MotorAmperageCondition` demonstrate the naming rule where the *Composition* `type` is converted to *Pascal Case* and then prepended to the *DataItem* `type`.

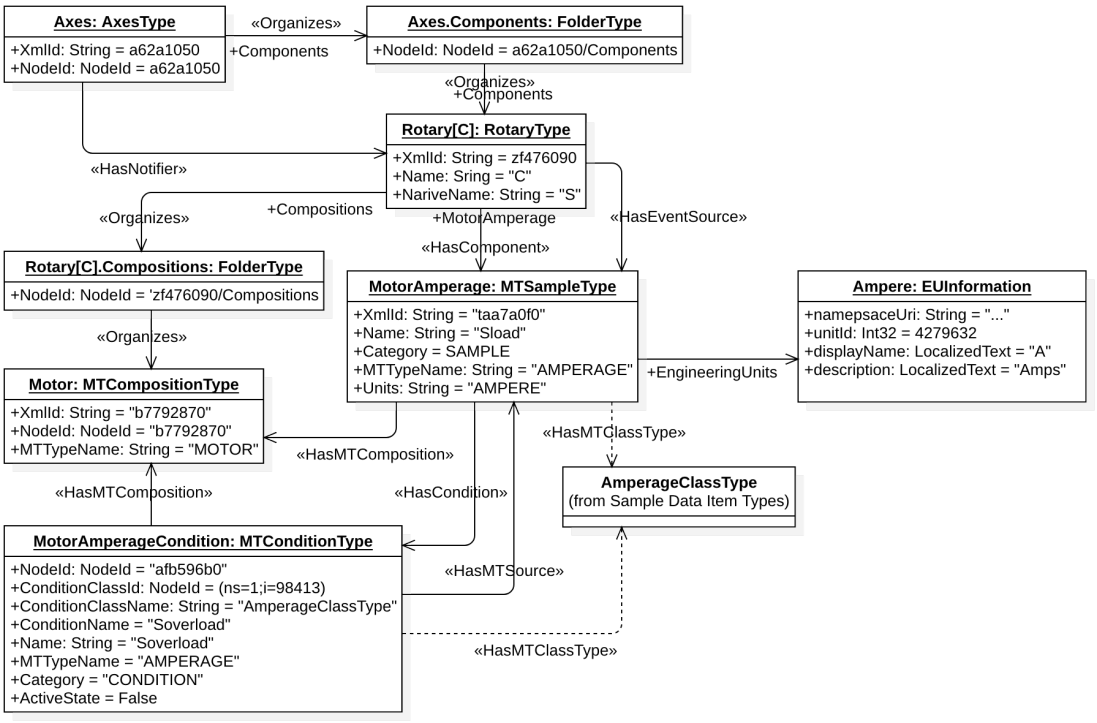

Figure 24: Rotary[C] Axis Motor Amperage DataItem

8.3.5.6 `Controller` and `Path` *Components*

In MTConnect, the `Controller` component represents the control system of the machine. This includes both the PLC and the Computer Numerical Controller (CNC) of the machine. In a CNC machine tool, the CNC and PLC are often thought of as a single component of the machine since they both control the operation of the machine.

The MTConnect information model does not differentiate between the two controllers, but represents them as a common controller category with data items that apply to both. Within a controller, there can be multiple `Paths`, each representing a set of coordinated motion. The `Path` is optional if the `Controller` is representing a simple machine with a PLC and no motion, such as a boiler or oven. This is covered in MTConnect Part 2 [MTConnect Part 2.0].

Listing 7: Controller and Path Components and Their Data Items

```
47      <Controller id="p5add360">
48        <DataItems>
49          <DataItem id="x7ca94e0" type="EMERGENCY_STOP"
    category="EVENT" name="estop"/>
50          <DataItem id="m17f1750" type="MESSAGE" category="
              EVENT"/>
```

```
        </DataItems>
        <Components>
          <Path id="a4a7bdf0" name="P1">
            <DataItems>
              <DataItem id="if36ff60" type="CONTROLLER_MODE"
    category="EVENT"/>
              <DataItem id="a01c7f30" type="EXECUTION" category
                  ="EVENT"/>
              <DataItem id="k8dd9030" type="PROGRAM" category="
                  EVENT"/>
              <DataItem id="r63f9b10" type="
                  CONTROLLER_MODE_OVERRIDE" subType="
                  OPTIONAL_STOP" category="EVENT"/>
              <DataItem id="a557d330" type="LOGIC_PROGRAM"
                  category="CONDITION"/>
              <DataItem id="a5b23650" type="MOTION_PROGRAM"
                  category="CONDITION"/>
              <DataItem id="bbafe670" type="LINE" category="
                  EVENT"/>
              <DataItem id="d2e9e4a0" type="PART_COUNT"
                  category="EVENT">
                <InitialValue>1</InitialValue>
              </DataItem>
              <DataItem id="r186cd60" type="PATH_POSITION"
                  category="SAMPLE" units="MILLIMETER_3D"/>
            </DataItems>
          </Path>
        </Components>
      </Controller>
```

The `Controller` has two `DataItems`, `Message` and `EmergencyStop` as shown in Figure 25. The `MTMessageType` is an **`Variable`** where the data type is a `MessageEventDataType`.

Figure 26 illustrates the `Path` component. Many of the *DataItems* types have been covered in prior examples, so the `Path` will only focus on the *DataItems* that have differentiated features. Those *DataItems* will be represented without additional properties or relations.

The `PATH_POSITION` is mapped to the `MTThreeSpaceSampleType` that represents a spacial coordinate in three space ($\mathbb{R}^3$) with each coordinate given in millimeters. The **`EngineeringUnits`** for this type will therefor always be mapped to **`MMT`** in the **`UNECE`** conventions.

The `MTThreeSpaceSampleType` uses the `ThreeSpacePositionDataType` that contains an *X, Y, and Z* **`Field`**. If one of the dimensions cannot be provided, it must be set to **`NaN`**.

The two `Conditions` in this component capable of branching. The model will use the **`BranchId`** in this case. This occurs when there are multiple simultaneous problems with

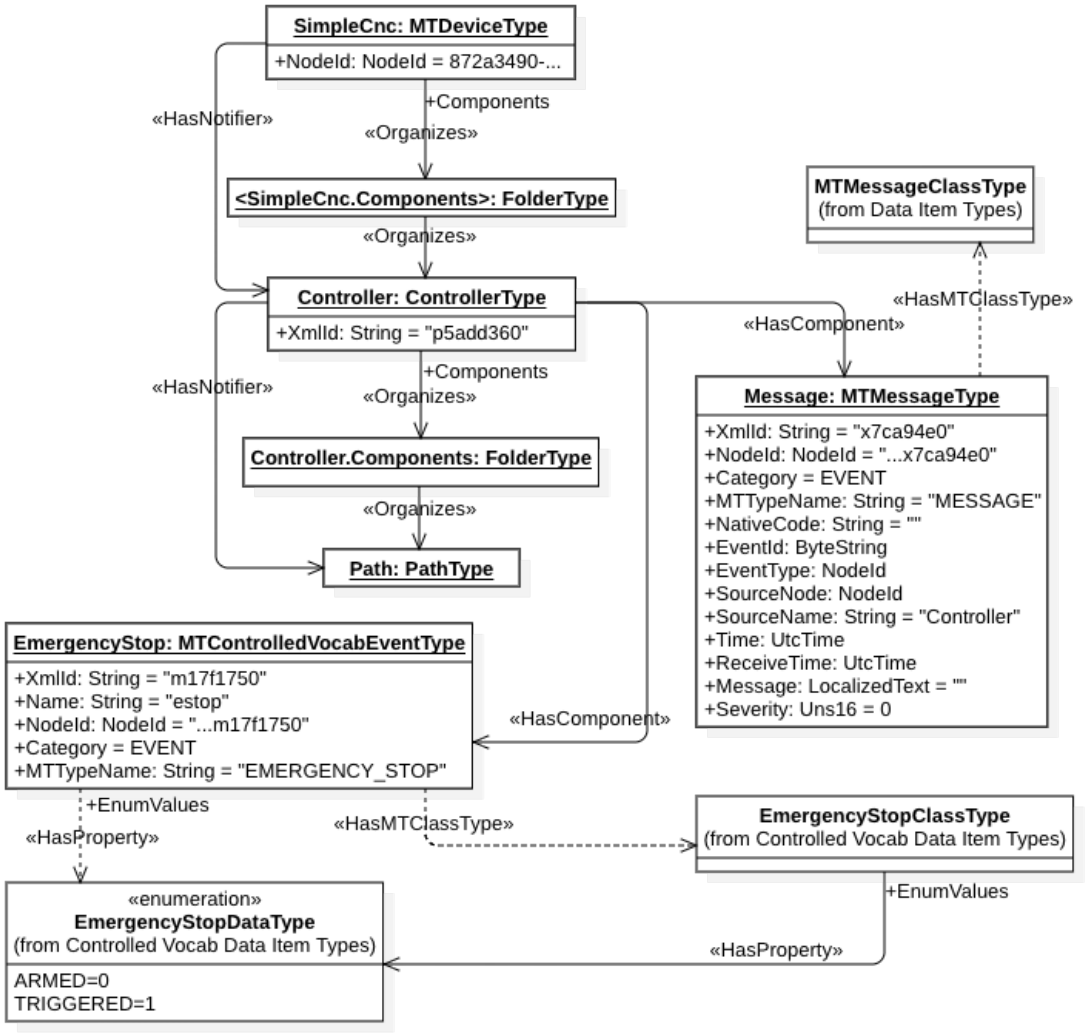

Figure 25: Controller Component and Data Items

either the logic or motion program in the controller. These cases will be discussed in the section on the streaming data.

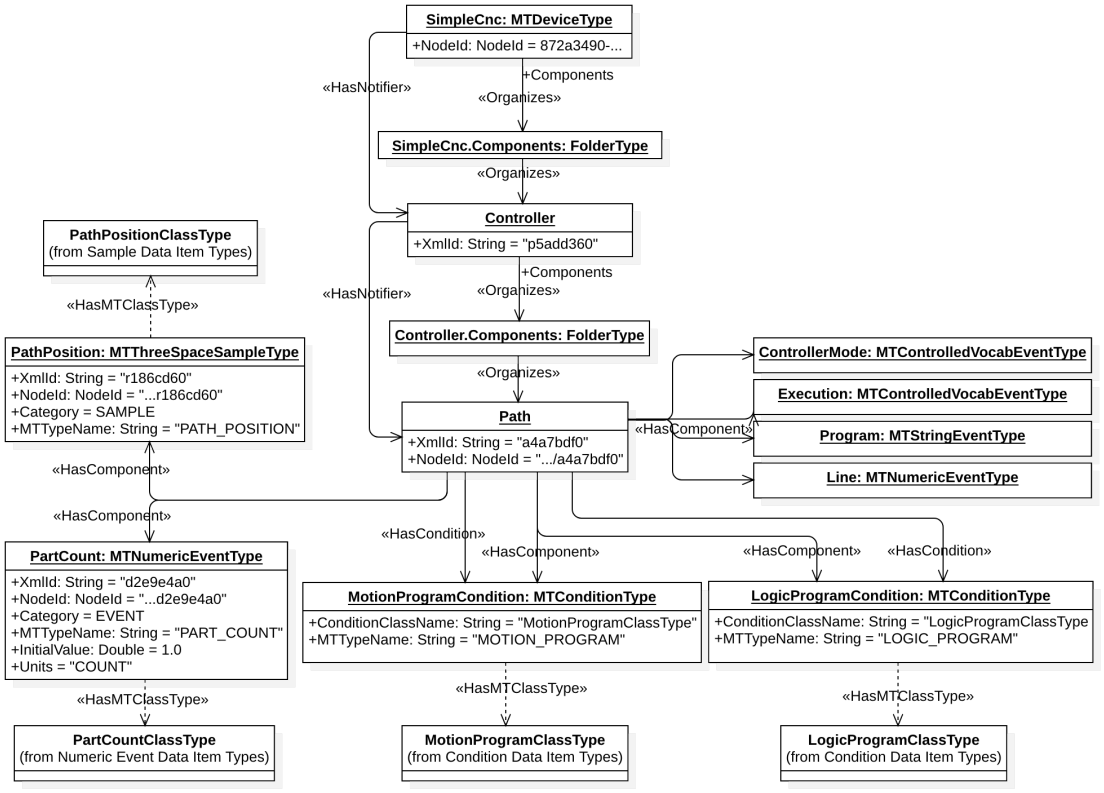

Figure 26: Path Component and Data Items

1242 8.3.5.7 `Electric` *System* of the `Device`

1243 The following example is the `Electric` system of the machine. The top level `Systems`
1244 component is an organizational component, similar to a **`Folder`** that collects *Compo-*
1245 *nents* that are central to a function in the device. The `Electric` component represents
1246 sensor data with some of the more advanced capabilities of the MTConnect standard. The
1247 `Electric` component demonstrate the use of a `Sensor` and *Configuration*.

1248

Listing 8: Electrical System and Sensor Configuration

1249
1250 70 `<Systems id="if618500">`
1251 71 `<Components>`
1252 72 `<Electric id="afb91ba0">`
1253 73 `<!-- Electric System Component First Set -->`
1254 74 `<DataItems>`
1255 75 `<DataItem id="x52ca7e0" name="ptemp" type="`
1256 `TEMPERATURE" category="SAMPLE" units="CELSIUS">`
1257 76 `<Source componentId="q9abfaf0"/>`
1258 77 `<Filters>`
1259 78 `<Filter type="PERIOD">60</Filter>`
1260 79 `</Filters>`

```
                </DataItem>
                <DataItem id="r1e58cf0" name="pvolt" type="
                    VOLTAGE" category="SAMPLE" units="VOLT">
                  <Filters>
                    <Filter type="MINIMUM_DELTA">10</Filter>
                  </Filters>
                </DataItem>
                <DataItem id="tc9edc70" name="pampts" type="
                    VOLT_AMPERE" category="SAMPLE" units="
                    VOLT_AMPERE" representation="TIME_SERIES"
                    sampleRate="100"/>

                <!-- Electric System Component Second Set -->
                <DataItem id="e25c1130" name="pamp" type="
                    AMPERAGE" category="SAMPLE" units="AMPERE"/>

                <DataItem id="qb9212c0" name="pampavg" type="
                    AMPERAGE" category="SAMPLE" units="AMPERE"
                    statistic="AVERAGE">
                  <ResetTrigger>ACTION_COMPLETE</ResetTrigger>
                </DataItem>
                <DataItem id="o63fcd30" name="ppfact" type="
                    POWER_FACTOR" category="SAMPLE" units="
                    PERCENT" />

                 <DataItem id="b4bb7110" type="AMPERAGE" category
                     ="CONDITION" name="poverload">
                   <Source dataItemId="e25c1130"/>
                 </DataItem>
                 <DataItem id="c82e32f0" type="TEMPERATURE"
                     category="CONDITION" name="povertemp">
                   <Source dataItemId="x52ca7e0"/>
                 </DataItem>
              </DataItems>
              <Components>
                <Sensor id="q9abfaf0">
                  <Configuration>
                    <SensorConfiguration>
                      <FirmwareVersion>23</FirmwareVersion>
                      <CalibrationDate>2018-08-12</
CalibrationDate>
                      <Channels>
                        <Channel number="1">
                          <Description>Temperature Probe</
                              Description>
                          <CalibrationDate>2018-09-11</
                              CalibrationDate>
                        </Channel>
                      </Channels>
                    </SensorConfiguration>
                  </Configuration>
                </Sensor>
              </Components>
            </Electric>
```

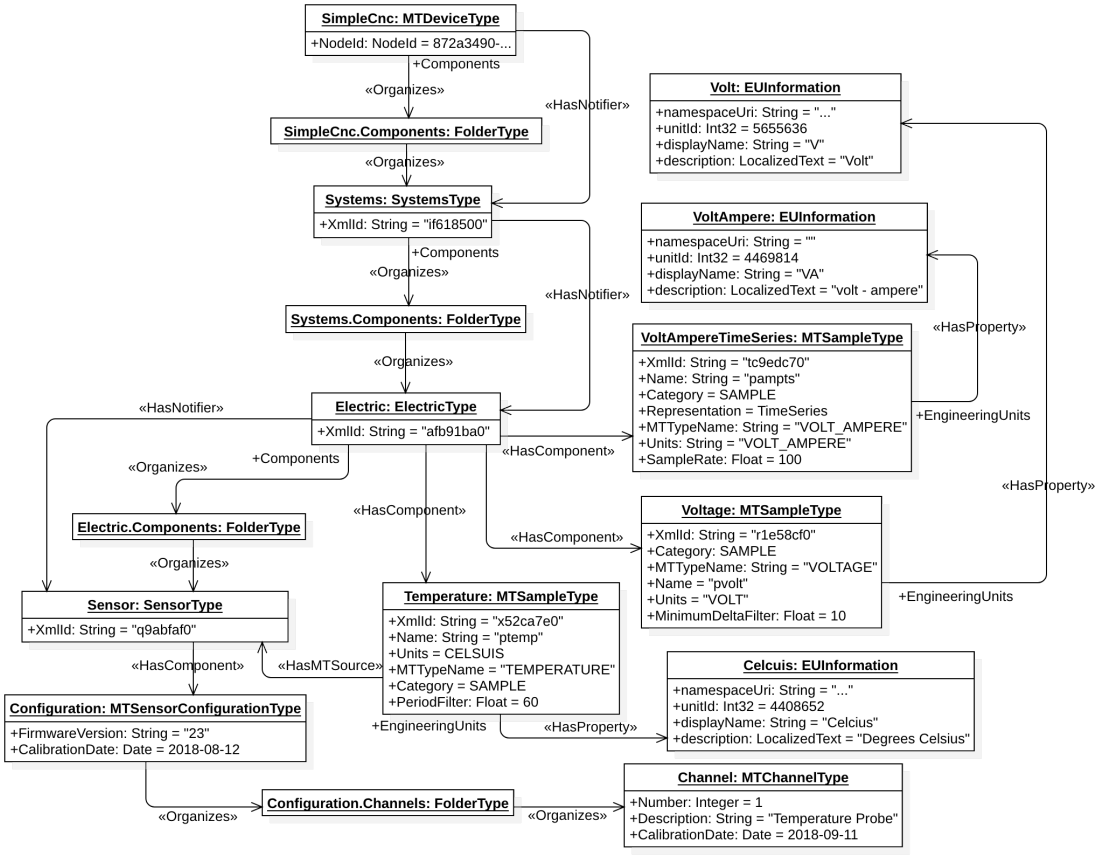

Figure 27: Electric System Component First Set

In the first set of *DataItems* from Listing 8 as shown in Figure 27, the first data item on line 75 has a period filter that is represented in the OPC UA ***Variable*** Temperature as a `PeriodFilter` ***Property*** with a value of 60.0. The period filters only reports changes after the period of time has elapsed, in this case 60 seconds–the `Temperature` will only publish changes every minute in this instance.

The `Temperature` ***Variable*** also has a `Source` element that references the `Sensor` with a `Configuration` of type `MTSensorConfigurationType`. The `Sensor-Configuration` has information about the `FirmwareVersion` and `CalibrationDate` of the `Sensor`. The `Channels` represent the multiple inputs to the `Sensor` unit. The first `Channel` has a `Description` and a `CalibrationDate` of the probe attached to the `Channel`. For more information on the `Channel`, see MTConnect Devices [MT-Connect Part 2.0].

The following *DataItem* on line 81 has a `Filter` using a `MINIMUM_DELTA` that is represened as a `MinimumDeltaFilter` in the UA ***Variable*** reprenenting the `MTSample-Type` Voltage. The `MinimumDelta` represents the smallest amount of change before the

change is reported. In this case the minimum change is 10 volts. Changes smaller than 10 volts will not be reported for this data item.

The second set of *DataItems* starting on line 89 shown in Figure 28 shows the relationship between the conditions and the other *DataItems*. As has been shown before the `Conditions` here are both related to a *DataItem* that is the source of the `Condition`. The *DataItem* therefor has a **`HasCondition`** relationship to the ***Condition*** and the *Component* has a **`HasEventSource`** ***Reference*** to the *DataItem*.

The `AverageAmperage` on line 90 has a `ResetTrigger`, specified using the `ResetTrigger` element having `CDATA ACTION_COMPLETE`, indicating that the `Average` statistic is taken only when the current `Action` has `Completed`, as defined by the process. This *DataItem* will only update on when the `ResetTrigger` fires.

Table 12 provides the OPC UA **`StatusCode`**s suggested values that need to be added in the working group to represent the MTConnect `ResetTrigger` model. The OPC Foundation TAC is asked to consider the addition of these additional status codes to support this capability in the MTConnect standard.

Table 12: MTConnect `ResetTrigger` to OPC UA **`StatusCode`** mapping

ResetTrigger	**StatusCode**	**Description**
ACTION_COMPLETE	**Good_ActionCompleteReset**	The value of the Data Entity that is measuring an action or operation was reset upon completion of that action or operation.
ANNUAL	**Good_AnnualReset**	The value of the Data Entity was reset at the end of a 12-month period.
DAY	**Good_DayReset**	The value of the Data Entity was reset at the end of a 24-hour period.
MAINTENANCE	**Good_MaintenanceReset**	The value of the Data Entity was reset upon completion of a maintenance event.
MANUAL	**Good_ManualReset**	The value of the Data Entity was reset based on a physical reset action.
MONTH	**Good_MonthReset**	The value of the Data Entity was reset at the end of a monthly period.
POWER_ON	**Good_PowerOnReset**	The value of the Data Entity was reset when power was applied to the piece of equipment after a planned or unplanned interruption of power has occurred.
SHIFT	**Good_ShiftReset**	The value of the Data Entity was reset at the end of a work shift.
WEEK	**Good_WeekReset**	The value of the Data Entity was reset at the end of a 7-day period.

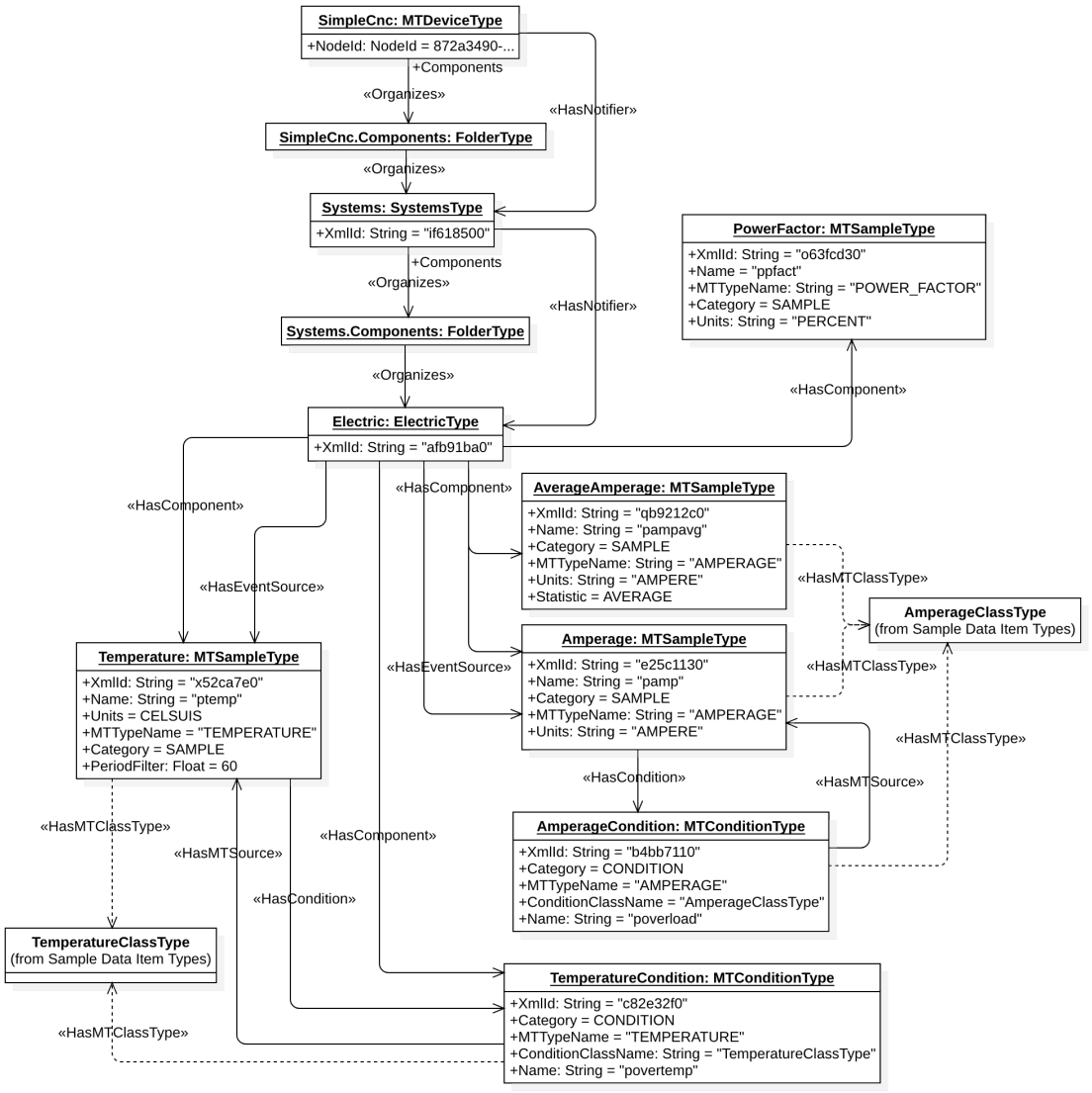

Figure 28: Electric System Component Second Set

1346 8.3.5.8 `Coolant` *System* with multiple *Tanks*

1347 The following example in Listing 9 demonstrates the situation where there are multiple
1348 components and data items of the same semantic type differentiated only by `name`. The
1349 machine has two `Coolant Systems`, one high pressure and one low pressure. Each of
1350 the two `Coolant Systems` has two tanks, a main and a reserve tank that are represented
1351 by *Compositions*.

1352

Listing 9: Coolant System and Sensor Configuration

1353
1354 119
```
<Coolant id="x5ef9730" name="low">
```

```
120   <DataItems>
121     <DataItem id="r25176b0" type="FILL_LEVEL" category="SAMPLE"
      units="PERCENT" name="low_main_level" compositionId="t59d1170
      "/>
122     <DataItem id="obc97840" type="FILL_LEVEL" category="SAMPLE"
          units="PERCENT" name="low_reserve_level" compositionId="
          a7973930"/>
123   </DataItems>
124   <Compositions>
125     <Composition id="t59d1170" type="TANK" name="main"/>
126     <Composition id="a7973930" type="TANK" name="reserve"/>
127   </Compositions>
128 </Coolant>
129 <Coolant id="b36e0070" name="high">
130   <DataItems>
131     <DataItem id="q94f81e0" type="FILL_LEVEL" category="SAMPLE"
      units="PERCENT" name="high_main_level" compositionId="
      a59bd5b0"/>
132     <DataItem id="wf2848e0" type="FILL_LEVEL" category="SAMPLE"
          units="PERCENT" name="high_reserve_level" compositionId="
          aa373750"/>
133   </DataItems>
134   <Compositions>
135     <Composition id="a59bd5b0" type="TANK" name="main"/>
136     <Composition id="aa373750" type="TANK" name="reserve"/>
137   </Compositions>
138 </Coolant>
```

Figure 29 presents the two coolant systems with names `high` and `low`. They are identical, each composed of two *Tank* compositions and each with two data items indication the *Fill Level* of the main and reserve *Tank*. The naming follows the rules for `TankFillLevel[low_main_level]` and `TankFillLevel[low_reserve_level]` for the `low` *Coolant System*.

The ***EngineeringUnits*** were not included in the figure as they have been covered before. The *Fill Level* `MTSampleType` will reference a *PERCENT* ***EUInformation Property*** as illustrated in the previous depiction of *Load* in Figure 23.

In this case the *Composition* name is converted to *PascalCase* and prepended to the *PascalCase* of the *DataItem* to create the base ***BrowseName***. Since there are two identical ***BrowseNames***, the `name` is appended in square brackets (`[]`). The same is done for the *Composition* **`BrowseName`** and the *Component* **`BrowseName`** "*Coolant*" giving `Coolant[high]` and `Coolant[low]`.

The two *Tank* compositions have similar treatment where they are named `Tank[main]` and `Tank[reserve]` for each of the two components. The *DataItems* then refer to each of the appropriate tanks to correctly reference the composition they are associated with. What is not show in the Figure 29 is the `FillLevelClassType` to save space. Each of the *DataItems* will have a `HasMTClassType` reference to the ***ClassType***.

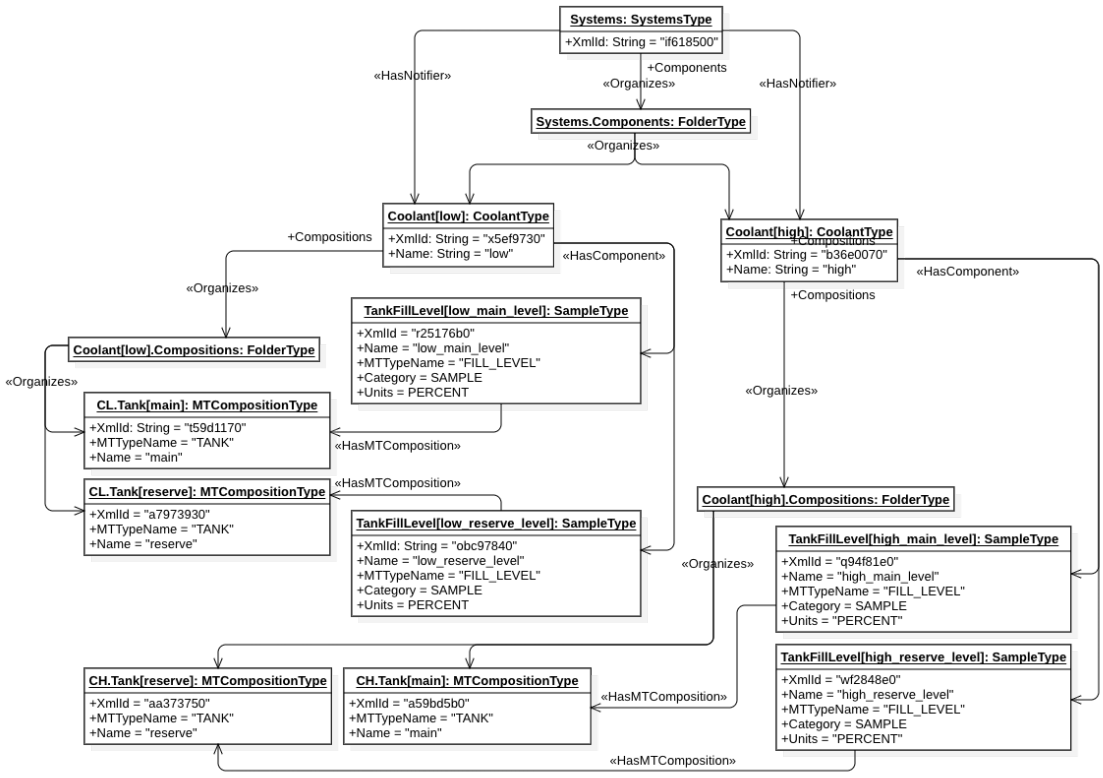

Figure 29: Coolant System

8.4 MTConnect Streaming Data

MTConnect separates streaming updates of data from the meta-data describing the semantic meaning of the data. [MTConnect Part 3.0] covers the streaming information model and the REST[Fie00] is covered in [MTConnect Part 1.0]. The protocol data in the `Header`, of the XML document, is ignored representing the data in OPC UA.

The HTTP REST Protocol provides two capabilities with respect to the collection of data that are common to most *publish/subscribe* technologies. They are as follows: first, a snapshot of the most recent know value for each *DataItem*, referred to as a *current request*, and second, a time-series of changes that occurred going back to the beginning of the store-and-forward `buffer`, referred to as a *sample request*.

Each entry in the *buffer* is time-stamped and assigned a sequence number that indicates the arrival order unique across all devices represented in the *Agent*. The sequence number is monotonically increasing and is unique concerning the instance (execution) of the *Agent*. When the *Agent* is restarted, the `sequence` may be reset to one indicating history has been expunged.

The following section provides the rules for taking data from the *MTConnectStream* XML Document and represent as a series of updates to the ***Variable*** and generate **`Events`** for

the MTConditionType and MTMessageType ***DataItem***.

8.4.1 MTConnectStreams Document Header

Listing 10: Streams Header

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <MTConnectStreams xmlns="urn:mtconnect.org:MTConnectStreams:1.4"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:
       schemaLocation="urn:mtconnect.org:MTConnectStreams:1.4 ./
       schema/MTConnectStreams_1.4.xsd">
3      <Header version="1.4.0" creationTime="2018-10-31T21:00:01
           Z" nextSequence="43124" lastSequence="44221"
           firstSequence="1" instanceId="1541045065" sender="
           localhost" bufferSize="131072"/>
4      <Streams>
```

Listing 10 is an example of the XML *Root Element* of the MTConnectStreams Document and contains the standard namespace declarations for MTConnect and XMLSchema. If additional namespaces are present, they must be added as additional **namespaces** in the OPC UA **Namespace** as well. The MTConnectStreams document begins with a Header containing protocol information that represents the version of the standard referenced and the store-and-forward *buffer* sequence numbers. [MTConnect Part 1.0] provides information on the interaction with the *Agent* and the proper sequence of requests to receive a contiguous stream of data from any point in the sequence of data.

8.4.2 MTConnectStreams Device and Component Stream

Listing 11: Component Stream

```
5      <ComponentStream componentId="e373fec0" component="Linear"
           name="X1" nativeName="X">
6    <DeviceStream name="SimpleCnc" uuid="872a3490-bd2d-0136-3eb0
       -0c85909298d9">
7      <Samples>
8        <Position sequence="131" timestamp="2018-10-31T20:33:11
   Z" dataItemId="dcbc0570" name="Xpos">UNAVAILABLE</Position>
9        <Position sequence="794" timestamp="2018-10-31T20:47:09
           Z" dataItemId="dcbc0570" name="Xpos">205.23</
           Position>
10     </Samples>
11     </ComponentStream>
```

MTConnect flattens the structure when reporting data and organizes it by *Device*, *Component* and category; this is shown in Listing 12 Line 5 with the DeviceStream,

ComponentStream, and Samples elements (not shown are the Events and Condition elements that will be address later). When the data stream is initialized or when it disconnects, the MTConnect *DataItem* Value is set to UNAVAILABLE as shown on Line 8.

MTConnect only reports data when it changes, and each change is assigned a monotonically increasing sequence number when it arrives at the MTConnect *Agent*. In addition to the sequence number; the XML attribute timestamp indicates the time the observation was made. The timestamp is used when setting the **Variable Time** along with the data and status. The data is only reported when it changes except for *DataItems* with the representation of DISCRETE indicating that each value has a discrete meaning, such as a PartCount where each count indicates an accrual of that many parts.

The value of UNAVAILABLE will be translated to the OPC UA **DataVariable StatusCode** of **Bad_NotConnected** to indicate that the value of the data is currently unknown from the data source.

The following examples provide the steps to translate from the MTConnectStreams document to the OPC UA information model presented above.

8.4.3 Samples

The *DataItems* with category SAMPLE are mapped to the MTSampleType which is a subtype of the **AnalogItemType DataVariable**. The numeric value of the sample sets the **Value** in the **AnalogItemType** ***Variable*** instance.

Listing 12: Linear Component Stream

```
12      <ComponentStream componentId="e373fec0" component="Linear"
            name="X1" nativeName="X">
13        <Samples>
14          <Position sequence="794" timestamp="2018-10-31T20
    :47:09.1011Z" dataItemId="dcbc0570" name="Xpos">205.23</
    Position>
15          <Position sequence="809" timestamp="2018-10-31T20
                :47:09.6021Z" dataItemId="dcbc0570" name="Xpos">
                206.23</Position>
16        </Samples>
17      </ComponentStream>
```

The OPC UA **DataVariableType**s are monitored items and have three attribtues that are updated when the value changes. They are the **Value** in the instance of the **DataVariable**, the **Time**, and the **Quality**. When the value is given, not UNAVAILABLE, the **Quality** will be set to the Status Code of **Good**, the **Time** will be set to the timestamp attribute and the **Value** set to the contents of the CDATA converted to a **Double**

precision value.

8.4.4 String and Numeric Events

The MTStringEventType and MTNumericEventType *DataItems* are managed in the same manner as the MTSampleType where the **Value**, **Quality**, and **Time** are taken from the CDATA, UNAVALAILABLE state, and timestamp respectively.

Listing 13: Path Component Stream

```
18      <ComponentStream componentId="a4a7bdf0" component="Path"
            name="P1">
19        <Events>
20          <Program sequence="430" timestamp="2018-10-31T20:47:09Z
        " dataItemId="k8dd9030">O98877</Program>
21          <PartCount sequence="630" timestamp="2018-10-31T20
                :57:09Z" dataItemId="d2e9e4a0">662</PartCount>
22          <ControllerMode sequence="255" timestamp="2018-10-31T20
                :27:09Z" dataItemId="if36ff60">AUTOMATIC</
                ControllerMode>
23        </Events>
24      </ComponentStream>
```

In Listing 13 on Line 20, the string value will be assigned to the *Program DataItem* in the Path component. The following Line 21 representing the PartCount will be converted to a numeric value, in this case an **Int32** and assigned to the **Value**.

All extended event types will be treated as MTStringEventTypes since it can support any value. It will be the responsibility of the application to interpret the value based on the extended **ClassType** associated with the **Variable**.

8.4.5 Controlled Vocabulary Events

Controlled Vocabularies are represented as MTControlledVocabEventType which is a subtype of the **MultiStateValueDiscreteType**. The **MultiStateValueDiscreteType** requires an integer index to be used for the **Value**, so the text given in MTConnect must be looked up in the **Ennumerations** in the **EnumValues** ***Property*** of the **ClassType** referenced by the *HasMTTypeClass* association. The index can be found by matching the CDATA to the **Fields** in the **Enumeration**.

Referring to Listing 13 Line 22, the values of the CDATA represent the allowed enumerations in the ControllerModeDataType as shown in Table 13. In this case the value from mtconnect is AUTOMATIC and is looked up in the associated **Enumera-**

tionDataType and the integer value is assigned to the **Value** of the MTControlled-VocabEventType instance, in this example the value will be 0 for AUTOMATIC.

Table 13: ControllerModeDataType Enumeration

Name	Index
AUTOMATIC	0
EDIT	1
MANUAL	2
MANUAL_DATA_INPUT	3
SEMI_AUTOMATIC	4

The **SourceTimestamp** and **Quality** will be handled as described in Section 8.4.4 for String and Numeric Events.

8.4.6 Conditions

Conditions are represented by the MTConditionType that is a subtype of the **ConditonType**. The MTConnect Conditions are a representation of the state of various alarms and health of a *Component* of the machine. There are three states for a condition in MTConnect, they are Normal, Warning, and Fault and have the semantic meaning *operating normally*, *a situation has been observed, but may self-correct*, and *a failure has occured and needs manual intervention*. More information can be found in MTConnect [MTConnect Part 2.0] and [MTConnect Part 3.0] of the MTConnect Standard for Condition modeling and behavior.

The MTConditionType has a ***Property*** called **ActiveState** that represents if there are any Warnings or Faults currently active. The **ActiveState** is an OPC UA **TwoStateVariableType Variable** defined in [UA Part 08]. When a Condition is Normal, the **ActiveState** is **False**, otherwise when either a Warning or Fault is present, the **ActiveState** is **True**.

Listing 14: Rotary C Component Stream

```
25      <ComponentStream componentId="zf476090" component="Rotary"
            name="C" nativeName="S">
26        <Condition>
27          <Normal sequence="201" timestamp="2018-10-31T20
        :34:19.9981Z" dataItemId="afb596b0" type="AMPERAGE"
        compositionId="b7792870" name="Soverload"/>
28          <Warning sequence="503" timestamp="2018-10-31T20
              :45:19.9981Z" dataItemId="afb596b0" type="AMPERAGE"
               compositionId="b7792870" name="Soverload"
              qualifier="HIGH" nativeCode="MOT-WARN">Spindle
              Motor Warning</Warning>
```

```
29          <Fault sequence="652" timestamp="2018-10-31T20
                :49:19.9981Z" dataItemId="afb596b0" type="AMPERAGE"
                 compositionId="b7792870" name="Soverload"
                qualifier="HIGH" nativeCode="MOT-OVR">Spindle Motor
                 Overload</Fault>
30      </Condition>
```

The instance of the MTConditionType will contain the last state of the ***Condition*** as most recently observed. Each time an MTConnect Condition activates or deactivates, an ***Condition*** **Event** will be created associated with the instance if the MTCondition-Type **NodeId**.

MTConditionType is a subtype of the **Event** and therefor is instantiated in the address space as an ***Object*** to maintain the **ActiveState** as well as each occurance of the Condition results in a instance of the **Event** being published as specified in [UA Part 03] and [UA Part 09].

8.4.6.1 Mapping Conditions

The ***ConditionType*** and **EventType** properties will be set as follows:

EnableState	• If the QName is Unavailable, **EnableState** is **False** • Otherwise, **EnableState** is **True**
Severity	• When Normal, **Severity** is 0. • When Warning, **Severity** is 500. • When Fault, **Severity** is 1000.
Retain	• When Normal, **False** • When Warning or Fault, **True**
Time	timestamp
NodeId	The **NodeId** of the **Event** is set to the **NodeId** of the MT-ConditionType ***Object*** instance.
Message	The CDATA of the Condition when Normal or Warning.
...	The remaining ***Properties*** will be set from the XML attributes of the same name.

8.4.6.2 MTConnect Condition Branching Example

MTConnect allows for multiple `Conditionss` to be active at the same time. This is similar to the concept of ***Branching*** in OPC UA. In MTConnect the the attribute `nativeCode` is used to indicate the association to the same ***Branch*** and is mapped to a branch id as illustrated in Figure 30. In this diagram there are three branches–0, 1, and 2–that are represent PLC alarms and are mapped to the `LOGIC_PROGRAM`. Each of these unique `nativeCodes` is mapped to a unique branch ID and is tacked separately. Only when the final condition transitions to normal, indicated by a *Normal* with no *nativeCode*, is the `Condition` considered cleared and completely inactive.

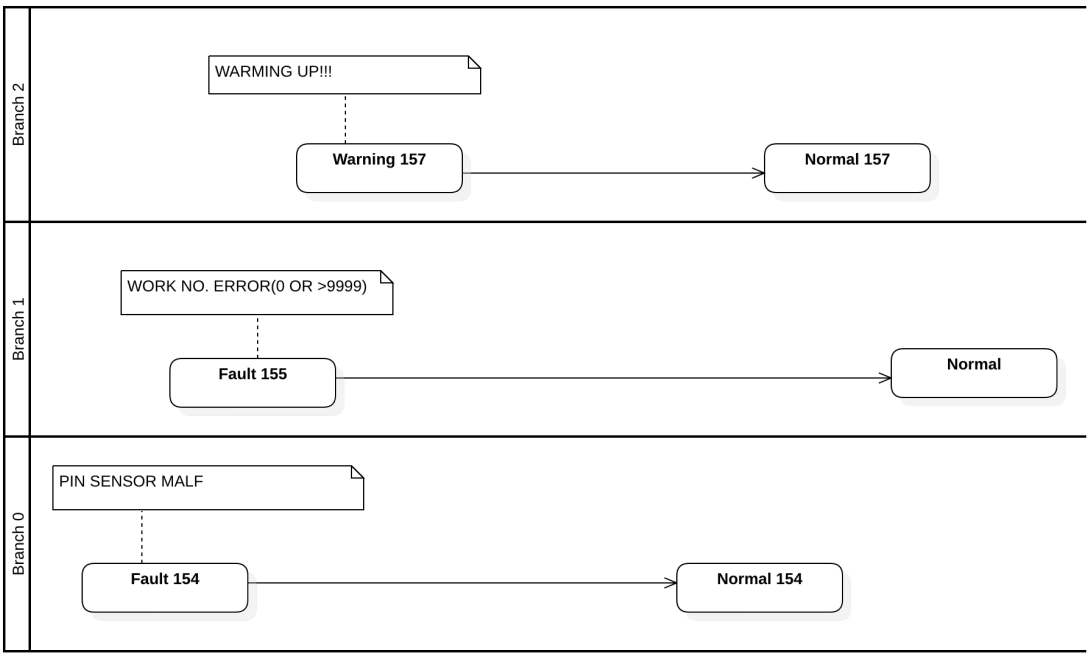

Figure 30: Condition Branching

Table 14 represents the state transitions of the key OPC UA **`Condition`** model and the `MTConditionType`. The text that follows will refer to this table and the MTConnect XML to illustrate the expected behavior.

MTConnect uses the `nativeCode` to determine the uniqueness of each activation of the Condition. If the *Path* component has a *DataItem* with a `type` of `LOGIC_PROGRAM` and there listing 7.

The initial state of the system is given in Table 14 Row 1. When the condition is inactive, the **`AciveState`** property is **`false`** and the **`Retain`** flag is also set to **`false`**. This corresponds to Listing 15 and the `Normal` initial condition state with no `nativeCode` indicating there are no active conditions.

Table 14: `LogicProgramCondition` States

Seq	Branch Id	Active	Retain	Native Code	Message
1	NULL	false	false	NULL	NULL
2	0	true	true	PLC–154	PIN SENSOR MALF
3	1	true	true	PLC–155	WORK NO. ERROR(0 OR >9999)
4	2	true	true	PLC–157	WARMING UP!!!
5	0	false	false	PLC–154	PIN SENSOR MALF
6	2	false	false	PLC–157	WARMING UP!!!
7	1	false	false	PLC–155	WORK NO. ERROR(0 OR >9999)
8	NULL	false	false	NULL	NULL

Listing 15: Path Logic Program Initial Normal State

```
<ComponentStream componentId="a4a7bdf0" component="Path" name="
    P1">
  <Condition>
    <Normal sequence="5200" timestamp="2018-10-31T20:30:19.9981
  Z" dataItemId="a557d330" type="LOGIC_PROGRAM"/>
  </Condition>
</ComponentStream>
```

The first fault occurred with the `nativeCode` `PLC-154` and creates the first branch as shown in Listing 16. If there is only one branch, then the **`BranchId`** will not be required and may remain **`NULL`**. In this example we are setting the **`BranchId`** to 1 as shown in Row 2 of Table 14. A `Fault` indicates a situation where the piece of equipment is no longer able to continue functioning and needs manual intervention.

Listing 16: Path Logic Program First Fault PLC-154

```
<ComponentStream componentId="a4a7bdf0" component="Path" name="
    P1">
  <Condition>
    <Fault sequence="5201" timestamp="2018-10-31T20:34:19.9981Z
  " dataItemId="a557d330" type="LOGIC_PROGRAM" nativeCode="PLC
  -154">PIN SENSOR MALF</Fault>
  </Condition>
</ComponentStream>
```

The second `Fault` is given in Listing 17 where a second PLC alarm is active. The native code is different than the previous condition, so a second branch must be created. The branch will assigned the number 1 for this instance and a OPC UA branch will be activated as illustrated in Row 3 of Table 14.

Listing 17: Path Logic Program Second Fault PLC-155

```
41  <ComponentStream componentId="a4a7bdf0" component="Path" name="
        P1">
42    <Condition>
43      <Fault sequence="5209" timestamp="2018-10-31T20:36:19.9981Z
      " dataItemId="a557d330" type="LOGIC_PROGRAM" nativeCode="PLC
      -155">WORK NO. ERROR(0 OR >9999)</Fault>
44    </Condition>
45  </ComponentStream>
```

The warning in Listing 18 indicates the machine is warming up and other operations are disabled. This condition has another `nativeCode` and therefor, like the previous condition, must create another branch. In this instance the **`BranchId`** will be 2. The `Warning` will be represented in UA as a **`severity`** and represents something that is of concern but not stopping the process. The warning is given by Row 4 of Table 14.

Listing 18: Path Logic Program Warning PLC-157

```
46  <ComponentStream componentId="a4a7bdf0" component="Path" name="
        P1">
47    <Condition>
48      <Warning sequence="5318" timestamp="2018-10-31T20
      :42:19.9981Z" dataItemId="a557d330" type="LOGIC_PROGRAM"
      nativeCode="PLC-157">WARMING UP!!!</Warning>
49    </Condition>
50  </ComponentStream>
```

In Listing 19, when the sensor malfunction is reset, the first condition will be returned to an inactive state. This is indicated by `Normal` and a native code of `PLC-154`. Since the other two conditions are still active, the `MTConditionType` is not inactive. In MTConnect, a `current` request would indicate that there is a Fault and a Warning currently active for this `Condition`. The clearing of this individual `Fault` is also represented on Row 5 of Table 14.

Listing 19: Path Logic Program Clear Fault of PLC-154

```
51  <ComponentStream componentId="a4a7bdf0" component="Path" name="
        P1">
52    <Condition>
53      <Normal sequence="5467" timestamp="2018-10-31T20:51:19.9981
      Z" dataItemId="a557d330" type="LOGIC_PROGRAM" nativeCode="PLC
      -154"/>
54    </Condition>
55  </ComponentStream>
```

In Listing 20, when the sensor malfunction is reset, the first condition will be returned to an

inactive state. It is indicated by `Normal` and a native code of `PLC-154`. Since the other two conditions are still active, the `MTConditionType` is not inactive. In MTConnect, a `current` request would indicate that there is a Fault and a Warning currently active for this `Condition`. Similar to the previous state, Table 14 clears the active state of this branch on Row 6.

Listing 20: Path Logic Program Clear Warning PLC-157

```
56  <ComponentStream componentId="a4a7bdf0" component="Path" name="
        P1">
57    <Condition>
58      <Normal sequence="5467" timestamp="2018-10-31T20:52:19.9981
      Z" dataItemId="a557d330" type="LOGIC_PROGRAM" nativeCode="PLC
      -157"/>
59    </Condition>
60  </ComponentStream>
```

Listing 21 represents the final `Normal` transition that clears all the currently active conditions and indicates that all the `Conditions` are now inactive or cleared and back to a `Normal` state. Row 7 of Table 14 shows the clearing of the final branch and then we clear everything in Row 8.

Listing 21: Path Logic Program Back to Normal, All Clear

```
61  <ComponentStream componentId="a4a7bdf0" component="Path" name="
        P1">
62    <Condition>
63      <Normal sequence="5467" timestamp="2018-10-31T20:57:19.9981
      Z" dataItemId="a557d330" type="LOGIC_PROGRAM"/>
64    </Condition>
65  </ComponentStream>
```

8.4.7 Messages

The `MTMessageType` is a data item with an extended type that uses the `MessageEventDataType` to provide the *NativeCode* and the *Text* of the message. The `CDATA` will be mapped to the *Text* field and the *NativeCode* will be mapped to the field of the UA **`DataType`** if it is provided as an `attribute`.

Listing 22: Path Motion Program Normal

```
66  <ComponentStream componentId="a4a7bdf0" component="Path" name="
        P1">
```

```
67      <Events>
68        <Message sequence="6241" timestamp="2018-10-31T20
          :37:19.9981Z" nativeCode="755" dataItemId="m17f1750">SELECT
          GRIPPED SURFACE</Message>
69        <Message sequence="6261" timestamp="2018-10-31T20
             :37:19.9981Z" naticeCode="866" dataItemId="m17f1750">
             SELECT TURNING SURFACE</Message>
70        <Message sequence="6422" timestamp="2018-10-31T20
             :37:19.9981Z" naticeCode="472" dataItemId="m17f1750">
             MEASURING STARTING POINT X</Message>
71        <Message sequence="6613" timestamp="2018-10-31T20
             :37:19.9981Z" naticeCode="996" dataItemId="m17f1750">
             MEASURING STARTING POINT Y</Message>
72      </Events>
73    </ComponentStream>
```

The MTMessageEventType is a sub-type of the **EventType** providing the message and native code associated with a given MTConnect message. The message source is the **Variable** MTMessageType that represents the meta data and the last message using the MessageDataType. The MessageDataType carries the nativeCode and the CDATA of the message text as represented in the MTConnectStreams document.

The representation allow for the association of meta data in the MTMessageType and the individual log of messages to be represented as **Event**s that can be queried and collected by the client application. The model is similar to the dual representation in the **Condition** types.

8.5 Time Series Samples

DataItems with TIME_SERIES representation provides a mechanism to publish high frequency data by combining multiple observations into a single vector of values. The attribute sampleRate is specified in the *Devices* information model if it is immutable or for each entity with the attribute sampleRate if it is variable.

In Listing 23, both examples are given, line 76 does not specify the sampleRate whereas line 78 specifies the sampleRate. The sampleRate is always given in hertz, if not specied in the MTConnectStreams document, it defaults to the value given in the *DataItem*.

Listing 23: Electric Component Time Series Example

```
74      <ComponentStream componentId="afb91ba0" component="Electric
           ">
75        <Samples>
76         <VoltAmpereTimeSeries sequence="1122" timestamp="
        2018-10-31T20:49:19.1981Z" dataItemId="tc9edc70" sampleCount=
```

```
    "10">421.23 422.36 419.55 420.14 421.98 422.32 418.25 419.75
    418.88 420.02</VoltAmpereTimeSeries>
77        <VoltAmpereTimeSeries sequence="1123" timestamp="
              2018-10-31T20:49:19.2981Z" dataItemId="tc9edc70"
              sampleCount="10">418.20 421.45 420.11 420.49 419.81
              419.06 417.54 420.53 417.67 421.48</
              VoltAmpereTimeSeries>
78        <VoltAmpereTimeSeries sequence="1124" timestamp="
              2018-10-31T20:49:19.3981Z" dataItemId="tc9edc70"
              sampleCount="10" sampleRate="100">418.09 420.48
              418.25 419.86 419.47 420.39 421.90 418.92 418.95
              420.73</VoltAmpereTimeSeries>
79        <VoltAmpereTimeSeries sequence="1125" timestamp="
              2018-10-31T20:49:19.4981Z" dataItemId="tc9edc70"
              sampleCount="10" sampleRate="100">420.27 419.63
              421.60 420.45 422.16 417.76 420.78 418.61 421.60
              418.04</VoltAmpereTimeSeries>
80      </Samples>
81    </ComponentStream>
```

In MTConnect, the *DataItem* element `QName` in the `MTConnectStreams` document appends the *Pascal Case* of the `representation` to the *Pascal Case* of the `type`. For example, `type` `VOLT_AMPERE` with the `representation` of `TIME_SERIES` becomes `VoltAmpereTimeSeries`.

All *DataItems* with `representation` of `TIME_SERIES` when reported in the `MTConnectStreams` document require the attribute `sampleCount` to give the number of entries in the space delimited series. MTConnect specifies in [MTConnect Part 3.0] that the `timestamp` is reported at the time of the last observation. To compute the `timestamp` for each observation, use the follow procedure.

$$\delta = \frac{1}{sampleRate}\ s \tag{1}$$

$$timestamp_1 = timestamp_a - (\delta \times sampleCount\ s) \tag{2}$$

$$timestamp_2 = timestamp_1 + \delta \tag{3}$$

$$timestamp_n = timestamp_1 + \delta \times n \tag{4}$$

Where $timestamp_a$ is the `timestamp` given as the XML Attribute.

When mapping *Time Series* to OPC UA, for each value in the series, compute the observation time as specified above and then update the UA **`Value`** with the **`Time`** set to the $timestamp_n$ for the n^{th} entry in the series. Do this once for every entry in the series causing `sampleCount` updates per MTConnect *Time Series* `Element`.

The δ in Equation (1) is the time interval for each sample and by subtracting the $\delta \times$ *sampleCount* seconds, the given in Equation (2) is the first timestamp in the series. Every

other timestamp can be derived by adding the $\delta \times n$ as showin in Equation (4) to the first timestamp. One can also arrive at the same result by taking the `timestamp` and counting backward from the last, where r is the reverse index starting at 0, $r \times \delta$ seconds from $timestamp_a$.

Time Series `timestamps` will often be contiguous when giving a stream of values for audio displacement or other series that must be reassembled. In Listing 23, each set of ten values are separated by 0.10 seconds which will allow for a 100hz sample frequence and continuous values from the sensor. The format is more efficient and allows for communication of waveform and high frequency data.

1812 9 MTConnect OPC UA Types

1813 9.1 Components

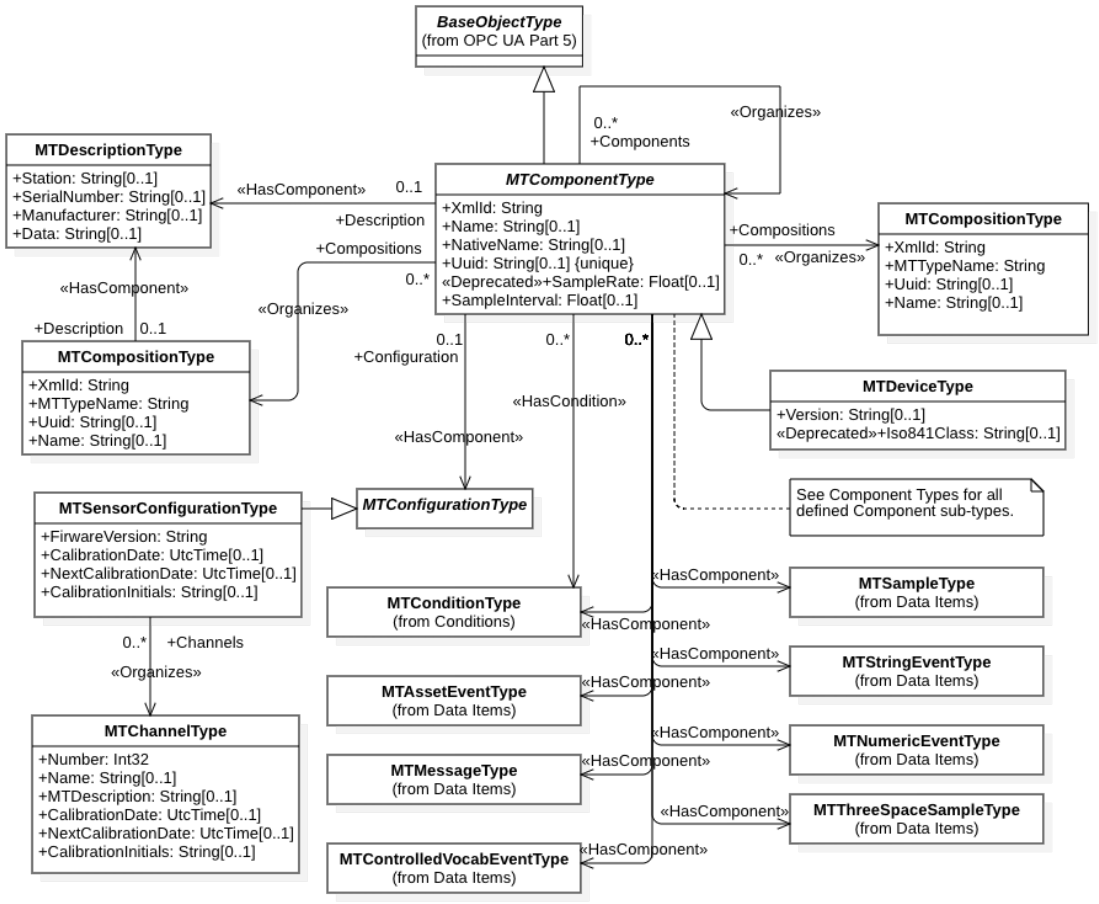

Figure 31: Components Diagram

1814 The *Components* documents the Component models and the owned objects.

1815 9.1.1 Defintion of `MTChannelType`

1816 A Channel of a sensor.

1817 See ChannelType in type specifications.

Table 15: `MTChannelType` Definition

Attribute	Value				
BrowseName	MTChannelType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of BaseObjectType (See [UA Part 05] 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

9.1.2 Defintion of `MTComponentType`

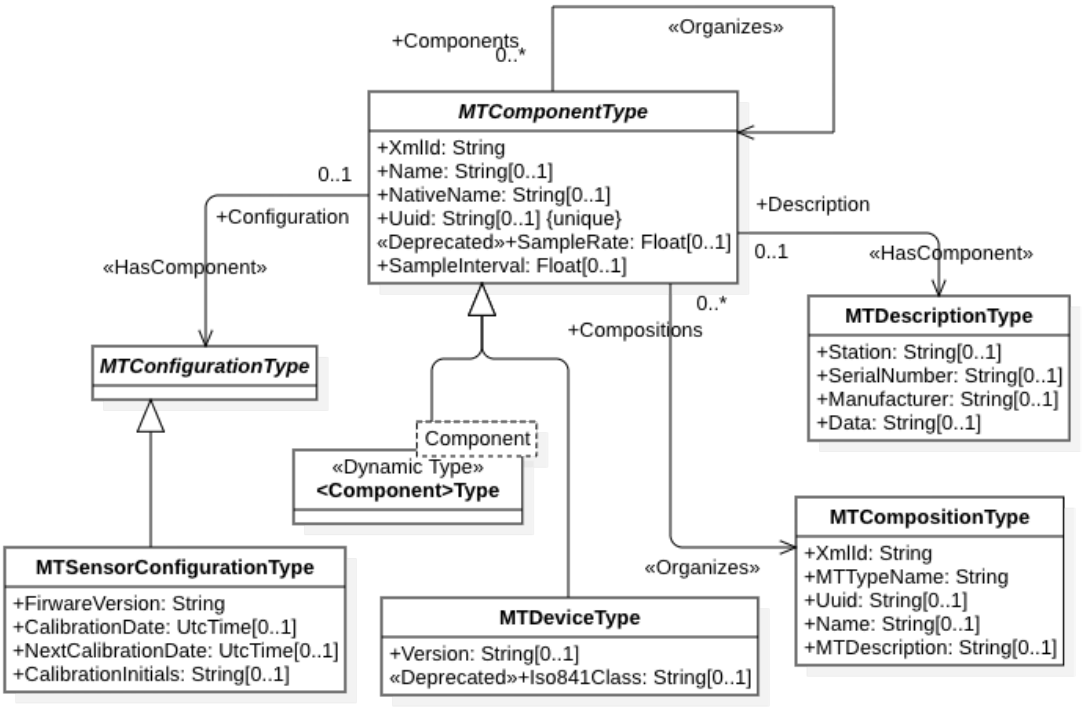

Figure 32: MTComponentType Diagram

The base *Component* Type from which all MTConnect Components are derived. The component types will be created once for all *Component* ***Objects*** of that type based on the `QName` of the MTConnect XML element.

The Component Objects will be created and inserted into the `Components` folder with a **`BrowseName`** of the Component `QName` and the `name` element if specified surrounded by square brackets, `[]`. For example if the MTConnect Element is:

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

The OPC UA Object with **`BrowseName`** `Linear[X]` will be created with the **`HasTypeDefinition`** referencing the `Linear` OPC UA ***Type***.

The meta data for the component and its relationships are static. The dynamic data will be represented using the [UA Part 08].

Table 16: `MTComponentType` Definition

Attribute	Value				
BrowseName	MTComponentType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**Type-Definition**	**Modeling-Rule**
Subtype of BaseObjectType (See [UA Part 05] Documentation)					
HasSubtype	ObjectType	MTDeviceType		See section 9.1.3	
HasSubtype	ObjectType	ActuatorType		See section 9.2.1	
HasSubtype	ObjectType	AuxiliariesType		See section 9.2.2	
HasSubtype	ObjectType	AxesType		See section 9.2.9	
HasSubtype	ObjectType	ControllerType		See section 9.2.13	
HasSubtype	ObjectType	DoorType		See section 9.2.15	
HasSubtype	ObjectType	InterfacesType		See section 9.2.16	
HasSubtype	ObjectType	ResourcesType		See section 9.2.21	
HasSubtype	ObjectType	SystemsType		See section 9.2.25	
HasProperty	Variable	XmlId	String	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	MTDescriptionType		Optional
HasComponent	Object	Configuration	MTConfigurationType		Optional
HasComponent	Variable	<MTControlledVocab-Event>	UInteger[]	MT-Controlled-VocabEvent-Type	Optional
HasComponent	Variable	<MTNumericEvent>	Number[]	MTNumeric-EventType	Optional
HasComponent	Variable	<MTStringEvent>	String[]	MTString-EventType	Optional
HasComponent	Variable	<MTSample>	Number[]	MTSample-Type	Optional
HasComponent	Variable	<MTMessage>	MessageDataType[]	MTMessage-Type	Optional
HasComponent	Variable	<MTAssetEvent>	AssetEventData-Type[]	MTAsset-EventType	Optional
HasComponent	Event	<MTCondition>	MTConditionType[]		Optional
HasCondition	Event	<MTCondition>	MTConditionType[]		Optional
HasComponent	Variable	<MTThreeSpaceSample>	ThreeSpaceSample-DataType[]	MTThree-SpaceSample-Type	Optional
Organizes	Object	Components	MTComponentType[]	FolderType	Optional
Organizes	Object	Compositions	MTComposition-Type[]	FolderType	Optional

9.1.3 Defintion of MTDeviceType

The MTDevice is a special type whose object will be the root of the device graph. The Device uses the component type factory and the component object factories to create each of the first level components.

The compositions, relationships, and data items are then recursively created as one descends the MTConnect information model.

Table 17: MTDeviceType Definition

Attribute	Value				
BrowseName	MTDeviceType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTComponentType (See section 9.1.2)					
HasProperty	Variable	Version	String	PropertyType	Optional
HasProperty	Variable	Iso841Class	String	PropertyType	Optional

9.1.3.1 Constraints

- Constraint uuid_not_empty:

```
1 uuid->notEmpty()
```

  Documentation: The UUID SHALL be provided.

- Constraint name_not_empty:

```
1 name->notEmpty()
```

  Documentation: The name of the Device SHALL be given.

9.1.4 Defintion of MTCompositionType

The MTCompositionType represents all composition entities. The specification of how to form the **BrowseName** is specified in Section 8.3.2.

The data items are added to the relationship where the *DataItem* to *Composition* relationship is represented by the **BrowseName** Composition property of the data item. The data items are added to the *Composition* by their **BrowseNames**.

Table 18: `MTCompositionType` Definition

Attribute	Value				
BrowseName	MTCompositionType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of BaseObjectType (See [UA Part 05] Documentation)					
HasProperty	Variable	XmlId	String	PropertyType	Mandatory
HasProperty	Variable	MTTypeName	String	PropertyType	Mandatory
HasProperty	Variable	Uuid	String	PropertyType	Optional
HasProperty	Variable	Name	String	PropertyType	Optional
HasComponent	Object	Description	MTDescriptionType		Optional

9.1.5 Defintion of `MTConfigurationType`

The abstract `MTConfigurationType` currently has only one sub-type, `MTSensorConfigurationType`. In the future, the configurations will also contain component and device configuration information as sub-types.

Table 19: `MTConfigurationType` Definition

Attribute	Value				
BrowseName	MTConfigurationType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of BaseObjectType (See [UA Part 05] Documentation)					
HasSubtype	ObjectType	MTSensorConfigurationType		See section 9.1.6	

9.1.6 Defintion of `MTSensorConfigurationType`

An MTConnect Sensor Configuration associated with the Component.

See SensorConfigurationType in type-specifications.

Table 20: `MTSensorConfigurationType` Definition

Attribute	Value				
BrowseName	MTSensorConfigurationType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTConfigurationType (See section 9.1.5)					
HasProperty	Variable	FirwareVersion	String	PropertyType	Mandatory
HasProperty	Variable	CalibrationDate	UtcTime	PropertyType	Optional
HasProperty	Variable	NextCalibrationDate	UtcTime	PropertyType	Optional
HasProperty	Variable	CalibrationInitials	String	PropertyType	Optional
Organizes	Object	Channels	MTChannelType[]	FolderType	Optional

1860 9.1.7 Defintion of `MTDescriptionType`

1861 An MTConnect Component Description.

1862 See the DescriptionType in the type-specifications.

Table 21: `MTDescriptionType` Definition

Attribute	Value				
BrowseName	MTDescriptionType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of BaseObjectType (See [UA Part 05] 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

1863 9.1.7.1 Referenced Properties and Objects

1864 - `Data::String`: From the CDATA of the Description Element in MTConnect.

1865 9.2 Component Types

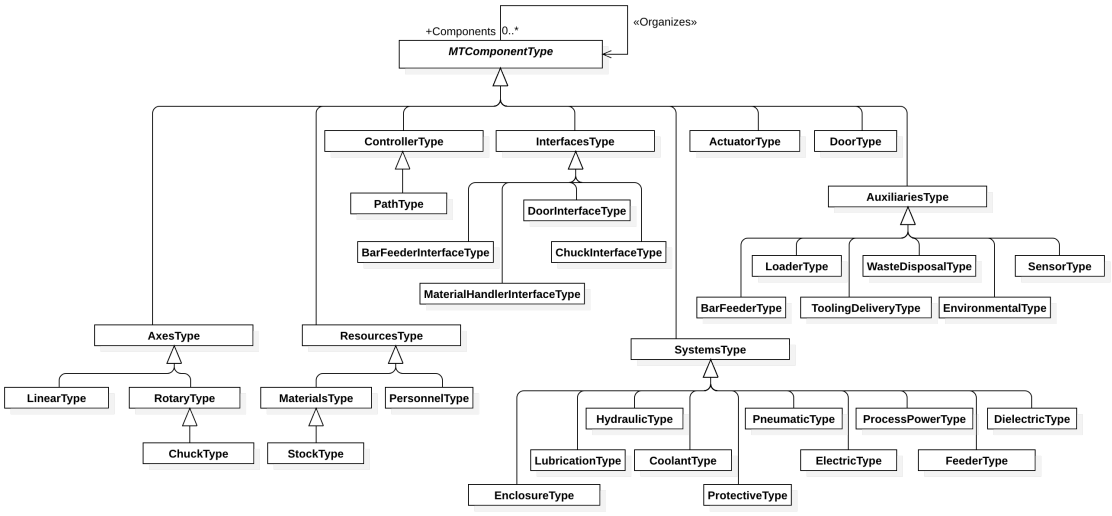

Figure 33: Component Types Diagram

All the sub types of components organized into top level organizational types.

9.2.1 Defintion of `ActuatorType`

the information for an apparatus for moving or controlling a mechanism or system

Table 22: `ActuatorType` Definition

Attribute	**Value**				
BrowseName	ActuatorType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTComponentType (See Components Documentation)					

9.2.2 Defintion of `AuxiliariesType`

representing functional sub-systems that provide supplementary or extended capabilities for a piece of equipment, but they are not required for the basic operation of the equipment

Table 23: `AuxiliariesType` Definition

Attribute	Value				
BrowseName	AuxiliariesType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTComponentType (See Components Documentation)					
HasSubtype	ObjectType	BarFeederType		See section 9.2.3	
HasSubtype	ObjectType	EnvironmentalType		See section 9.2.4	
HasSubtype	ObjectType	LoaderType		See section 9.2.5	
HasSubtype	ObjectType	SensorType		See section 9.2.6	
HasSubtype	ObjectType	ToolingDeliveryType		See section 9.2.7	
HasSubtype	ObjectType	WasteDisposalType		See section 9.2.8	

1872 9.2.3 Defintion of `BarFeederType`

1873 a unit involved in delivering bar stock to a piece of equipment.

Table 24: `BarFeederType` Definition

Attribute	Value				
BrowseName	BarFeederType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of AuxiliariesType (See section 9.2.2)					

1874 9.2.4 Defintion of `EnvironmentalType`

1875 the information for a unit or function involved in monitoring, managing, or conditioning
1876 the environment around or within a piece of equipment.

Table 25: `EnvironmentalType` Definition

Attribute	Value				
BrowseName	EnvironmentalType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of AuxiliariesType (See section 9.2.2)					

1877 9.2.5 Defintion of `LoaderType`

1878 the information for a unit comprised of all the parts involved in moving and distributing
1879 materials, parts, tooling, and other items to or from a piece of equipment

Table 26: `LoaderType` Definition

Attribute	Value				
BrowseName	LoaderType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of AuxiliariesType (See section 9.2.2)					

1880 9.2.6 Defintion of `SensorType`

1881 the information for a piece of equipment that responds to a physical stimulus and transmits
1882 a resulting impulse or value from a sensing unit

Table 27: `SensorType` Definition

Attribute	Value				
BrowseName	SensorType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of AuxiliariesType (See section 9.2.2)					

1883 9.2.7 Defintion of `ToolingDeliveryType`

1884 a unit involved in managing, positioning, storing, and delivering tooling within a piece of
1885 equipment.

Table 28: `ToolingDeliveryType` Definition

Attribute	Value				
BrowseName	ToolingDeliveryType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of AuxiliariesType (See section 9.2.2)					

1886 9.2.8 Defintion of `WasteDisposalType`

1887 the information for a unit comprised of all the parts involved in removing manufacturing
1888 byproducts from a piece of equipment

Table 29: `WasteDisposalType` Definition

Attribute	Value				
BrowseName	WasteDisposalType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of AuxiliariesType (See section 9.2.2)					

1889 9.2.9 Defintion of `AxesType`

1890 Organizes parts of the device that perform linear or rotational motion

Table 30: `AxesType` Definition

Attribute	Value				
BrowseName	AxesType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTComponentType (See Components Documentation)					
HasSubtype	ObjectType	LinearType		See section 9.2.10	
HasSubtype	ObjectType	RotaryType		See section 9.2.11	

1891 9.2.10 Defintion of `LinearType`

1892 the movement of a physical piece of equipment, or a portion of the equipment, in a straight
1893 line.

Table 31: `LinearType` Definition

Attribute	Value				
BrowseName	LinearType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of AxesType (See section 9.2.9)					

1894 9.2.11 Defintion of `RotaryType`

1895 rotary movement of a physical piece of equipment or a portion of the equipment.

Table 32: `RotaryType` Definition

Attribute	Value				
BrowseName	RotaryType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of AxesType (See section 9.2.9)					
HasSubtype	ObjectType	ChuckType		See section 9.2.12	

1896 9.2.12 Defintion of `ChuckType`

1897 provides the information about a mechanism that holds a part or stock material in place

Table 33: `ChuckType` Definition

Attribute	Value				
BrowseName	ChuckType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of RotaryType (See section 9.2.11)					

1898 9.2.13 Defintion of `ControllerType`

1899 intelligent or computational function within a piece of equipment

Table 34: `ControllerType` Definition

Attribute	Value				
BrowseName	ControllerType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTComponentType (See Components Documentation)					
HasSubtype	ObjectType	PathType		See section 9.2.14	

1900 9.2.14 Defintion of `PathType`

1901 information for an independent operation or function within a `ControllerType`

Table 35: `PathType` Definition

Attribute	Value				
BrowseName	PathType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of ControllerType (See section 9.2.13)					

9.2.15 Defintion of `DoorType`

the information for a mechanical mechanism or closure that can cover, for example, a physical access portal into a piece of equipment

Table 36: `DoorType` Definition

Attribute	Value				
BrowseName	DoorType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTComponentType (See Components Documentation)					

9.2.16 Defintion of `InterfacesType`

Table 37: `InterfacesType` Definition

Attribute	Value				
BrowseName	InterfacesType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTComponentType (See Components Documentation)					
HasSubtype	ObjectType	BarFeederInterfaceType		See section 9.2.17	
HasSubtype	ObjectType	ChuckInterfaceType		See section 9.2.18	
HasSubtype	ObjectType	DoorInterfaceType		See section 9.2.19	
HasSubtype	ObjectType	MaterialHandlerInterfaceType		See section 9.2.20	

9.2.17 Defintion of `BarFeederInterfaceType`

information used to coordinate the operations between a Bar Feeder and another piece of equipment

Table 38: `BarFeederInterfaceType` Definition

Attribute	Value				
BrowseName	BarFeederInterfaceType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of InterfacesType (See section 9.2.16)					

1909 9.2.18 Defintion of `ChuckInterfaceType`

1910 information used to coordinate the operations between two pieces of equipment, one of
1911 which controls the operation of a chuck

Table 39: `ChuckInterfaceType` Definition

Attribute	Value				
BrowseName	ChuckInterfaceType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of InterfacesType (See section 9.2.16)					

1912 9.2.19 Defintion of `DoorInterfaceType`

1913 information used to coordinate the operations between two pieces of equipment, one of
1914 which controls the operation of a door

Table 40: `DoorInterfaceType` Definition

Attribute	Value				
BrowseName	DoorInterfaceType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of InterfacesType (See section 9.2.16)					

1915 9.2.20 Defintion of `MaterialHandlerInterfaceType`

1916 set of information used to coordinate the operations between a piece of equipment and
1917 another associated piece of equipment used to automatically handle various types of ma-
1918 terials or services associated with the original piece of equipment

Table 41: `MaterialHandlerInterfaceType` Definition

Attribute	Value				
BrowseName	MaterialHandlerInterfaceType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of InterfacesType (See section 9.2.16)					

1919 9.2.21 Defintion of `ResourcesType`

Table 42: `ResourcesType` Definition

Attribute	Value				
BrowseName	ResourcesType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTComponentType (See Components Documentation)					
HasSubtype	ObjectType	MaterialsType		See section 9.2.22	
HasSubtype	ObjectType	PersonnelType		See section 9.2.24	

1920 9.2.22 Defintion of `MaterialsType`

1921 information about materials or other items consumed or used by the piece of equipment
1922 for production of parts, materials, or other types of goods

Table 43: `MaterialsType` Definition

Attribute	Value				
BrowseName	MaterialsType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of ResourcesType (See section 9.2.21)					
HasSubtype	ObjectType	StockType		See section 9.2.23	

1923 9.2.23 Defintion of `StockType`

1924 the information for the material that is used in a manufacturing process and to which work
1925 is applied in a machine or piece of equipment to produce parts.

Table 44: `StockType` Definition

Attribute	Value				
BrowseName	StockType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MaterialsType (See section 9.2.22)					

9.2.24 Defintion of `PersonnelType`

Table 45: `PersonnelType` Definition

Attribute	Value				
BrowseName	PersonnelType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of ResourcesType (See section 9.2.21)					

9.2.25 Defintion of `SystemsType`

major sub-systems that are permanently integrated into a piece of equipment

Table 46: `SystemsType` Definition

Attribute	Value				
BrowseName	SystemsType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTComponentType (See Components Documentation)					
HasSubtype	ObjectType	CoolantType		See section 9.2.26	
HasSubtype	ObjectType	DielectricType		See section 9.2.27	
HasSubtype	ObjectType	ElectricType		See section 9.2.28	
HasSubtype	ObjectType	EnclosureType		See section 9.2.29	
HasSubtype	ObjectType	FeederType		See section 9.2.30	
HasSubtype	ObjectType	HydraulicType		See section 9.2.31	
HasSubtype	ObjectType	LubricationType		See section 9.2.32	
HasSubtype	ObjectType	PneumaticType		See section 9.2.33	
HasSubtype	ObjectType	ProcessPowerType		See section 9.2.34	
HasSubtype	ObjectType	ProtectiveType		See section 9.2.35	

9.2.26 Defintion of CoolantType

a system comprised of all the parts involved in distribution and management of fluids that remove heat from a piece of equipment.

Table 47: CoolantType Definition

Attribute	Value				
BrowseName	CoolantType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of SystemsType (See section 9.2.25)					

9.2.27 Defintion of DielectricType

a system that manages a chemical mixture used in a manufacturing process being performed at that piece of equipment.

Table 48: DielectricType Definition

Attribute	Value				
BrowseName	DielectricType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of SystemsType (See section 9.2.25)					

9.2.28 Defintion of ElectricType

represents the information for the main power supply for device piece of equipment and the distribution of that power throughout the equipment.

Table 49: ElectricType Definition

Attribute	Value				
BrowseName	ElectricType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of SystemsType (See section 9.2.25)					

9.2.29 Defintion of EnclosureType

a structure used to contain or isolate a piece of equipment or area.

Table 50: `EnclosureType` Definition

Attribute	Value				
BrowseName	EnclosureType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of SystemsType (See section 9.2.25)					

9.2.30 Defintion of `FeederType`

the information for a system that manages the delivery of materials within a piece of equipment.

Table 51: `FeederType` Definition

Attribute	Value				
BrowseName	FeederType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of SystemsType (See section 9.2.25)					

9.2.31 Defintion of `HydraulicType`

system comprised of all the parts involved in moving and distributing pressurized liquid throughout the piece of equipment.

Table 52: `HydraulicType` Definition

Attribute	Value				
BrowseName	HydraulicType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of SystemsType (See section 9.2.25)					

9.2.32 Defintion of `LubricationType`

a system comprised of all the parts involved in distribution and management of fluids used to lubricate portions of the piece of equipment.

Table 53: `LubricationType` Definition

Attribute	Value				
BrowseName	LubricationType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of SystemsType (See section 9.2.25)					

1949 9.2.33 Defintion of `PneumaticType`

1950 a system comprised of all the parts involved in moving and distributing pressurized gas
1951 throughout the piece of equipment.

Table 54: `PneumaticType` Definition

Attribute	Value				
BrowseName	PneumaticType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of SystemsType (See section 9.2.25)					

1952 9.2.34 Defintion of `ProcessPowerType`

1953 the information for a power source associated with a piece of equipment that supplies
1954 energy to the manufacturing process separate from the Electric system

Table 55: `ProcessPowerType` Definition

Attribute	Value				
BrowseName	ProcessPowerType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of SystemsType (See section 9.2.25)					

1955 9.2.35 Defintion of `ProtectiveType`

1956 the information for those functions that detect or prevent harm or damage to equipment or
1957 personnel.

Table 56: `ProtectiveType` Definition

Attribute	Value				
BrowseName	ProtectiveType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of SystemsType (See section 9.2.25)					

9.3 Data Items

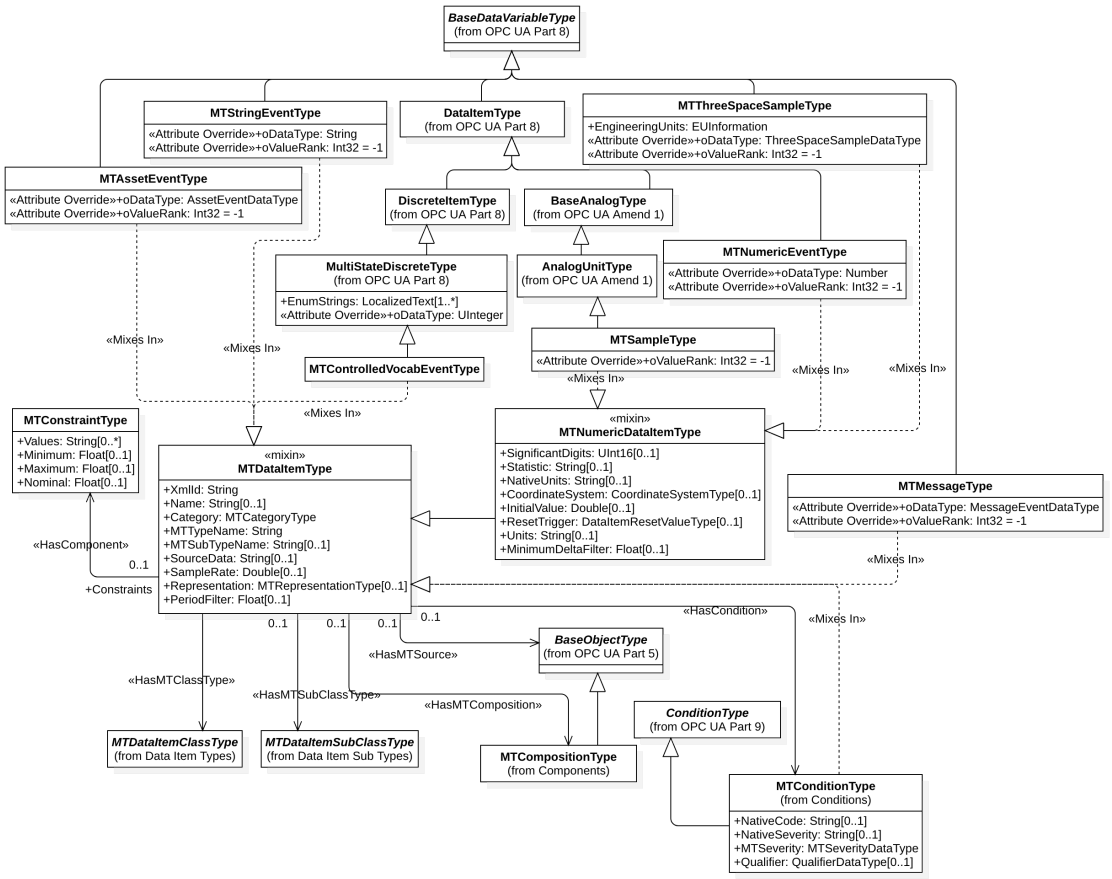

Figure 34: Data Items Diagram

DataItems in MTConnect reprst data that can be reported by a piece of manufacturing equipment and are mapped to the ***DataVariableTypes*** in OPC UA. The mapping rules are given in Section 8 and leverage the capabilities of the OPC UA standard to represent data reported by manufacturing equipment. There are some MTConnect *DataItems* that are represented using mechanisms beyond the ***DataVariableType***, these are the *Condtions category* and the *Messages type*s.

All the *DataItems* of with a *category* `CONDITION` and are mapped to the `MTCondi-`

tionType that is decended from the **Event** base types. The *Message* is also an **Event** type, but does not retain an active state as the Condition does.

The MTConnect *DataItem* has a lot of contextual meta-data that is used in each of the MTConnect types that are decended from different OPC UA **DataVariable** to make greatest use of the OPC UA standard [UA Part 08]. The Companion specifiction uses a *«mixin»* pattern that provides a set of common ***Properties*** and ***Components*** that are included in all sub-types that represent MTConnect *DataItems*.

The following types all mixin the common properties from the MTDataItemType or MTNumericDataItemType depending on the nature of the information.

9.3.1 Defintion of AssetEventDataType

A special ***Variable*** data type for asset change with a AssetType and AssetId.

Table 57: AssetEventDataType DataType

Field	Type	Optional
AssetId	String	Mandatory
AssetType	String	Mandatory

9.3.2 Defintion of MTAssetEventType

The asset events have an additional attribute regarding the asset change or removal identifier and the type of asset that is being reported.

9.3.2.1 Dependencies and Relationships

- Mixes in MTDataItemType, see See section 9.3.6

Table 58: `MTAssetEventType` Definition

Attribute	Value				
BrowseName	MTAssetEventType				
IsAbstract	False				
ValueRank	-1				
DataType	AssetEventDataType				
References	**NodeClass**	**BrowseName**	**DataType**	**Type-Definition**	**Modeling-Rule**
Subtype of BaseDataVariableType (See [UA Part 08] Documentation)					
HasProperty	Variable	XmlId	String	PropertyType	Mandatory
HasProperty	Variable	Name	String	PropertyType	Optional
HasProperty	Variable	Category	MTCategoryType	PropertyType	Mandatory
HasProperty	Variable	MTTypeName	String	PropertyType	Mandatory
HasProperty	Variable	MTSubTypeName	String	PropertyType	Optional
HasProperty	Variable	SourceData	String	PropertyType	Optional
HasProperty	Variable	SampleRate	Double	PropertyType	Optional
HasProperty	Variable	Representation	MTRepresentation-Type	PropertyType	Optional
HasProperty	Variable	PeriodFilter	Float	PropertyType	Optional
HasMTClassType	Object	<MTDataItemClass>	MTDataItemClassType		Mandatory
HasMTSubClass-Type	Object	<MTDataItemSub-Class>	MTDataItemSubClassType		Optional
HasMT-Composition	Object	<MTComposition>	MTCompositionType		Optional
HasMTSource	Object	<BaseObject>	BaseObjectType		Optional
HasCondition	Event	<MTCondition>	MTConditionType		Optional
HasComponent	Object	Constraints	MTConstraintType		Optional

9.3.3 Defintion of `MTConditionClassType`

The abstract type for all data items types that are specifically for `CONDITION` `category`.

Table 59: `MTConditionClassType` Definition

Attribute	Value				
BrowseName	MTConditionClassType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemClassType (See Data Item Types Documentation)					
HasSubtype	ObjectType	ActuatorClassType		See section 9.10.1	
HasSubtype	ObjectType	CommunicationsClassType		See section 9.10.2	
HasSubtype	ObjectType	DataRangeClassType		See section 9.10.3	
HasSubtype	ObjectType	LogicProgramClassType		See section 9.10.5	
HasSubtype	ObjectType	HardwareClassType		See section 9.10.4	
HasSubtype	ObjectType	MotionProgramClassType		See section 9.10.6	
HasSubtype	ObjectType	SystemClassType		See section 9.10.7	

1985 9.3.4 Defintion of `MTConstraintType`

1986 The MTConnect constraints. The Values or the Minimum, Maximum, and Nominal values
1987 should be provided. Multiple Values can be provided as an array as a set of allowable
1988 values for this *DataItem*.

Table 60: `MTConstraintType` Definition

Attribute	Value				
BrowseName	MTConstraintType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of BaseObjectType (See [UA Part 05] Documentation)					
HasProperty	Variable	Values	String[]	PropertyType	Optional
HasProperty	Variable	Minimum	Float	PropertyType	Optional
HasProperty	Variable	Maximum	Float	PropertyType	Optional
HasProperty	Variable	Nominal	Float	PropertyType	Optional

1989 9.3.5 Defintion of `MTControlledVocabEventType`

1990 All *DataItems* with `category EVENT` having Controlled Vocabularies (Enumerations)
1991 will be added as sub-types of this type which is mapped to the OPC/UA MultiStateVal-
1992 ueDiscreteType. Otherwise, either `MTString` or `MTNumeric` will be used. All subtypes
1993 are direct representations of the MTConnect equivalent elements that can be found in the
1994 MTConnect Part 3 [MTConnect Part 3.0] documents.

Table 61: `MTControlledVocabEventType` Definition

Attribute	Value				
BrowseName	MTControlledVocabEventType				
IsAbstract	False				
ValueRank	-2				
DataType	UInteger				
References	**NodeClass**	**BrowseName**	**DataType**	**Type-Definition**	**Modeling-Rule**
Subtype of MultiStateDiscreteType (See [UA Part 08] Documentation)					
HasProperty	Variable	XmlId	String	PropertyType	Mandatory
HasProperty	Variable	Name	String	PropertyType	Optional
HasProperty	Variable	Category	MTCategoryType	PropertyType	Mandatory
HasProperty	Variable	MTTypeName	String	PropertyType	Mandatory
HasProperty	Variable	MTSubTypeName	String	PropertyType	Optional
HasProperty	Variable	SourceData	String	PropertyType	Optional
HasProperty	Variable	SampleRate	Double	PropertyType	Optional
HasProperty	Variable	Representation	MTRepresentation-Type	PropertyType	Optional
HasProperty	Variable	PeriodFilter	Float	PropertyType	Optional
HasMTClassType	Object	<MTDataItemClass>	MTDataItemClassType		Mandatory
HasMTSubClass-Type	Object	<MTDataItemSub-Class>	MTDataItemSubClassType		Optional
HasMT-Composition	Object	<MTComposition>	MTCompositionType		Optional
HasMTSource	Object	<BaseObject>	BaseObjectType		Optional
HasCondition	Event	<MTCondition>	MTConditionType		Optional
HasComponent	Object	Constraints	MTConstraintType		Optional

9.3.5.1 Dependencies and Relationships

- Mixes in `MTDataItemType`, see See section 9.3.6

9.3.6 Defintion of «mixin» MTDataItemType

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

Table 62: `MTDataItemType` Definition

Attribute	Value				
BrowseName	MTDataItemType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**Type-Definition**	**Modeling-Rule**
HasSubtype	ObjectType	MTNumericDataItemType		See section 9.3.7	
HasProperty	Variable	XmlId	String	PropertyType	Mandatory
HasProperty	Variable	Name	String	PropertyType	Optional
HasProperty	Variable	Category	MTCategoryType	PropertyType	Mandatory
HasProperty	Variable	MTTypeName	String	PropertyType	Mandatory
HasProperty	Variable	MTSubTypeName	String	PropertyType	Optional
HasProperty	Variable	SourceData	String	PropertyType	Optional
HasProperty	Variable	SampleRate	Double	PropertyType	Optional
HasProperty	Variable	Representation	MTRepresentation-Type	PropertyType	Optional
HasProperty	Variable	PeriodFilter	Float	PropertyType	Optional
HasMTClassType	Object	<MTDataItemClass>	MTDataItemClassType		Mandatory
HasMTSubClass-Type	Object	<MTDataItemSub-Class>	MTDataItemSubClassType		Optional
HasMT-Composition	Object	<MTComposition>	MTCompositionType		Optional
HasMTSource	Object	<BaseObject>	BaseObjectType		Optional
HasCondition	Event	<MTCondition>	MTConditionType		Optional
HasComponent	Object	Constraints	MTConstraintType		Optional

9.3.6.1 Referenced Properties and Objects

- **Allowable Values** for `MTCategoryType`

  Represents the `category` attribute of the MTConnect *DataItem*.

Table 63: `MTCategoryType` Enumeration

Name	Index
EVENT	0
CONDITION	1
SAMPLE	2

- `SourceData::String:` The text that is the `CDATA` of the `Source` element.
- **Allowable Values** for `MTRepresentationType`

  Represents the `representation` attribute of the MTConnect *DataItem*.

Table 64: `MTRepresentationType` Enumeration

Name	Index
DISCRETE	0
TIME_SERIES	1
VALUE	2

- `PeriodFilter::Float:` Represents the MTConnect `Filter` subtype of `PeriodFilter`.

- **Allowable Values** for `MTRepresentationType`

  Represents the `representation` attribute of the MTConnect *DataItem*.

Table 65: `MTRepresentationType` Enumeration

Name	Index
DISCRETE	0
TIME_SERIES	1
VALUE	2

- **Allowable Values** for `MTCategoryType`

  Represents the `category` attribute of the MTConnect *DataItem*.

Table 66: `MTCategoryType` Enumeration

Name	Index
EVENT	0
CONDITION	1
SAMPLE	2

9.3.6.2 Operations

- `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.

9.3.6.3 Dependencies and Relationships

- **Allowable Values** for `MTRepresentationType`

  Represents the `representation` attribute of the MTConnect *DataItem*.

Table 67: `MTRepresentationType` Enumeration

Name	Index
DISCRETE	0
TIME_SERIES	1
VALUE	2

- **Allowable Values** for `MTCategoryType`

  Represents the `category` attribute of the MTConnect *DataItem*.

Table 68: `MTCategoryType` Enumeration

Name	Index
EVENT	0
CONDITION	1
SAMPLE	2

9.3.7 Defintion of «mixin» MTNumericDataItemType

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

Table 69: `MTNumericDataItemType` Definition

Attribute	Value				
BrowseName	MTNumericDataItemType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modeling-Rule**
Subtype of MTDataItemType (See section 9.3.6)					
HasProperty	Variable	SignificantDigits	UInt16	PropertyType	Optional
HasProperty	Variable	Statistic	MTStatisticType	PropertyType	Optional
HasProperty	Variable	NativeUnits	String	PropertyType	Optional
HasProperty	Variable	CoordinateSystem	MTCoordinateSystem-Type	PropertyType	Optional
HasProperty	Variable	InitialValue	Double	PropertyType	Optional
HasProperty	Variable	ResetTrigger	MTResetTriggerType	PropertyType	Optional
HasProperty	Variable	Units	String	PropertyType	Optional
HasProperty	Variable	MinimumDeltaFilter	Float	PropertyType	Optional

9.3.7.1 Referenced Properties and Objects

- `SignificantDigits::UInt16`: The significant digits is converted to a ***ValuePrecision***.
- **Allowable Values** for `MTStatisticType`

Table 70: `MTStatisticType` Enumeration

Name	Index
AVERAGE	0
MAXIMUM	1
MEDIAN	2
MINIMUM	3
MODE	4
RANGE	5
ROOT_MEAN_SQUARE	6
STANDARD_DEVIATION	7

- **Allowable Values** for `MTCoordinateSystemType`

  Represents the `coordinateSystem` attribute of the MTConnect *DataItem*.

Table 71: `MTCoordinateSystemType` Enumeration

Name	Index
MACHINE	0
WORK	1

- **Allowable Values** for `MTResetTriggerType`

  These need to become **`Good_`** status code in OPC UA.

Table 72: `MTResetTriggerType` Enumeration

Name	Index
ACTION_COMPLETE	0
ANNUAL	1
DAY	2
MAINTENANCE	3
MANUAL	4
MONTH	5
POWER_ON	6
SHIFT	7
WEEK	8

- `MinimumDeltaFilter::Float:` Represents the MTConnect `Filter` subtype of `MinimumDeltaFilter`.

9.3.8 Defintion of `MTEventClassType`

The base type class for all data items with a `category` of `EVENT`.

Table 73: `MTEventClassType` Definition

Attribute	Value				
BrowseName	MTEventClassType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemClassType (See Data Item Types Documentation)					
HasSubtype	ObjectType	MTControlledVocabEventClassType		See section 9.7.1	
HasSubtype	ObjectType	MTNumericEventClassType		See section 9.8.1	
HasSubtype	ObjectType	MTStringEventClassType		See section 9.9.1	
HasSubtype	ObjectType	MTMessageClassType		See section 9.5.2	

9.3.9 Defintion of MTMessageEventType

Table 74: MTMessageEventType Definition

Attribute	Value				
BrowseName	MTMessageEventType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of BaseEventType (See [UA Part 05] Documentation)					
HasProperty	Variable	NativeCode	String	PropertyType	Optional

9.3.10 Defintion of MTMessageType

The message is a sub-type of the ***DataVariableType*** using the MessageEventDataType to represent the values for *NativeCode* and *Text* of the message from the CDATA of the MTConnect Streams message.

9.3.10.1 Dependencies and Relationships

- Mixes in MTDataItemType, see See section 9.3.6

Table 75: `MTMessageType` Definition

Attribute	Value				
BrowseName	MTMessageType				
IsAbstract	False				
ValueRank	-1				
DataType	MessageDataType				
References	**NodeClass**	**BrowseName**	**DataType**	**Type-Definition**	**Modeling-Rule**
Subtype of BaseDataVariableType (See [UA Part 08] Documentation)					
HasProperty	Variable	XmlId	String	PropertyType	Mandatory
HasProperty	Variable	Name	String	PropertyType	Optional
HasProperty	Variable	Category	MTCategoryType	PropertyType	Mandatory
HasProperty	Variable	MTTypeName	String	PropertyType	Mandatory
HasProperty	Variable	MTSubTypeName	String	PropertyType	Optional
HasProperty	Variable	SourceData	String	PropertyType	Optional
HasProperty	Variable	SampleRate	Double	PropertyType	Optional
HasProperty	Variable	Representation	MTRepresentation-Type	PropertyType	Optional
HasProperty	Variable	PeriodFilter	Float	PropertyType	Optional
HasMTClassType	Object	<MTDataItemClass>	MTDataItemClassType		Mandatory
HasMTSubClass-Type	Object	<MTDataItemSub-Class>	MTDataItemSubClassType		Optional
HasMT-Composition	Object	<MTComposition>	MTCompositionType		Optional
HasMTSource	Object	<BaseObject>	BaseObjectType		Optional
HasCondition	Event	<MTCondition>	MTConditionType		Optional
HasComponent	Object	Constraints	MTConstraintType		Optional

9.3.11 Defintion of `MTNumericEventType`

All data items with category `Event` and a numeric value. These are usually counters for parts and lines. Currently only builtin types that are known to be integers will be sub-typed from this type. Extended types will be subtyped from the `MTStringEventType`.

9.3.11.1 Dependencies and Relationships

- Mixes in `MTNumericDataItemType`, see See section 9.3.7

Table 76: `MTNumericEventType` Definition

Attribute	Value				
BrowseName	MTNumericEventType				
IsAbstract	False				
ValueRank	-1				
DataType	Number				
References	**NodeClass**	**BrowseName**	**DataType**	**Type-Definition**	**Modeling-Rule**
Subtype of DataItemType (See [UA Part 08] Documentation)					
HasProperty	Variable	XmlId	String	PropertyType	Mandatory
HasProperty	Variable	Name	String	PropertyType	Optional
HasProperty	Variable	Category	MTCategoryType	PropertyType	Mandatory
HasProperty	Variable	MTTypeName	String	PropertyType	Mandatory
HasProperty	Variable	MTSubTypeName	String	PropertyType	Optional
HasProperty	Variable	SourceData	String	PropertyType	Optional
HasProperty	Variable	SampleRate	Double	PropertyType	Optional
HasProperty	Variable	Representation	MTRepresentation-Type	PropertyType	Optional
HasProperty	Variable	PeriodFilter	Float	PropertyType	Optional
HasMTClassType	Object	<MTDataItemClass>	MTDataItemClassType		Mandatory
HasMTSubClass-Type	Object	<MTDataItemSub-Class>	MTDataItemSubClassType		Optional
HasMT-Composition	Object	<MTComposition>	MTCompositionType		Optional
HasMTSource	Object	<BaseObject>	BaseObjectType		Optional
HasCondition	Event	<MTCondition>	MTConditionType		Optional
HasComponent	Object	Constraints	MTConstraintType		Optional
HasProperty	Variable	SignificantDigits	UInt16	PropertyType	Optional
HasProperty	Variable	Statistic	MTStatisticType	PropertyType	Optional
HasProperty	Variable	NativeUnits	String	PropertyType	Optional
HasProperty	Variable	CoordinateSystem	MTCoordinate-SystemType	PropertyType	Optional
HasProperty	Variable	InitialValue	Double	PropertyType	Optional
HasProperty	Variable	ResetTrigger	MTResetTrigger-Type	PropertyType	Optional
HasProperty	Variable	Units	String	PropertyType	Optional
HasProperty	Variable	MinimumDeltaFilter	Float	PropertyType	Optional

9.3.12 Defintion of MTSampleType

Data Items with category SAMPLE. The simplest mapping since all these types are floating point numeric data and comply with the **AnalogUnitType** from [UA Part 08] Amendment 1. In ammendment 1, the **EURange** is optional. **EngineeringUnits** for all MTSampleType Data Items.

The **EURange** will becreated if the Constraints element exists and both Maximum and Minimum values are given.

Table 77: `MTSampleType` Definition

Attribute	Value				
BrowseName	MTSampleType				
IsAbstract	False				
ValueRank	-1				
DataType	Number				
References	**NodeClass**	**BrowseName**	**DataType**	**Type-Definition**	**Modeling-Rule**
Subtype of AnalogUnitType (See [UA Amend 1] Documentation)					
HasProperty	Variable	XmlId	String	PropertyType	Mandatory
HasProperty	Variable	Name	String	PropertyType	Optional
HasProperty	Variable	Category	MTCategoryType	PropertyType	Mandatory
HasProperty	Variable	MTTypeName	String	PropertyType	Mandatory
HasProperty	Variable	MTSubTypeName	String	PropertyType	Optional
HasProperty	Variable	SourceData	String	PropertyType	Optional
HasProperty	Variable	SampleRate	Double	PropertyType	Optional
HasProperty	Variable	Representation	MTRepresentation-Type	PropertyType	Optional
HasProperty	Variable	PeriodFilter	Float	PropertyType	Optional
HasMTClassType	Object	<MTDataItemClass>	MTDataItemClassType		Mandatory
HasMTSubClass-Type	Object	<MTDataItemSub-Class>	MTDataItemSubClassType		Optional
HasMT-Composition	Object	<MTComposition>	MTCompositionType		Optional
HasMTSource	Object	<BaseObject>	BaseObjectType		Optional
HasCondition	Event	<MTCondition>	MTConditionType		Optional
HasComponent	Object	Constraints	MTConstraintType		Optional
HasProperty	Variable	SignificantDigits	UInt16	PropertyType	Optional
HasProperty	Variable	Statistic	MTStatisticType	PropertyType	Optional
HasProperty	Variable	NativeUnits	String	PropertyType	Optional
HasProperty	Variable	CoordinateSystem	MTCoordinate-SystemType	PropertyType	Optional
HasProperty	Variable	InitialValue	Double	PropertyType	Optional
HasProperty	Variable	ResetTrigger	MTResetTrigger-Type	PropertyType	Optional
HasProperty	Variable	Units	String	PropertyType	Optional
HasProperty	Variable	MinimumDeltaFilter	Float	PropertyType	Optional

9.3.12.1 Referenced Properties and Objects

- `Supertype::AnalogUnitType:` All DataItems with category SAMPLE

9.3.12.2 Dependencies and Relationships

- Mixes in `MTNumericDataItemType`, see See section 9.3.7

9.3.13 Defintion of `MTStringEventType`

All data items with `category` **`EVENT`** where the data is freeform text. The data type will be set to String for all the sub-types. All extended type, regardless of controlled vocabularies, will use this base type unless proprietary enumerations are added to the nodeset as required by the builtin state event types inherited from `MTControlledVocabEvent-Type` (see 9.3.5).

9.3.13.1 Dependencies and Relationships

- Mixes in `MTDataItemType`, see See section 9.3.6

Table 78: `MTStringEventType` Definition

Attribute	Value				
BrowseName	MTStringEventType				
IsAbstract	False				
ValueRank	-1				
DataType	String				
References	**NodeClass**	**BrowseName**	**DataType**	**Type-Definition**	**Modeling-Rule**
Subtype of BaseDataVariableType (See [UA Part 08] Documentation)					
HasProperty	Variable	XmlId	String	PropertyType	Mandatory
HasProperty	Variable	Name	String	PropertyType	Optional
HasProperty	Variable	Category	MTCategoryType	PropertyType	Mandatory
HasProperty	Variable	MTTypeName	String	PropertyType	Mandatory
HasProperty	Variable	MTSubTypeName	String	PropertyType	Optional
HasProperty	Variable	SourceData	String	PropertyType	Optional
HasProperty	Variable	SampleRate	Double	PropertyType	Optional
HasProperty	Variable	Representation	MTRepresentation-Type	PropertyType	Optional
HasProperty	Variable	PeriodFilter	Float	PropertyType	Optional
HasMTClassType	Object	<MTDataItemClass>	MTDataItemClassType		Mandatory
HasMTSubClass-Type	Object	<MTDataItemSub-Class>	MTDataItemSubClassType		Optional
HasMT-Composition	Object	<MTComposition>	MTCompositionType		Optional
HasMTSource	Object	<BaseObject>	BaseObjectType		Optional
HasCondition	Event	<MTCondition>	MTConditionType		Optional
HasComponent	Object	Constraints	MTConstraintType		Optional

9.3.14 Defintion of `MTThreeSpaceSampleType`

A special data item type that represents a three space coordinate. It uses a data type with three fields, X, Y, and Z, where the coordinates are given in millimeters. The EngineeringUnits will always be set to MMT in the UNECE convetion.

9.3.14.1 Dependencies and Relationships

- Mixes in `MTNumericDataItemType`, see See section 9.3.7

Table 79: `MTThreeSpaceSampleType` Definition

Attribute	Value				
BrowseName	MTThreeSpaceSampleType				
IsAbstract	False				
ValueRank	-1				
DataType	ThreeSpaceSampleDataType				
References	**NodeClass**	**BrowseName**	**DataType**	**Type-Definition**	**Modeling-Rule**
Subtype of BaseDataVariableType (See [UA Part 08] Documentation)					
HasProperty	Variable	XmlId	String	PropertyType	Mandatory
HasProperty	Variable	Name	String	PropertyType	Optional
HasProperty	Variable	Category	MTCategoryType	PropertyType	Mandatory
HasProperty	Variable	MTTypeName	String	PropertyType	Mandatory
HasProperty	Variable	MTSubTypeName	String	PropertyType	Optional
HasProperty	Variable	SourceData	String	PropertyType	Optional
HasProperty	Variable	SampleRate	Double	PropertyType	Optional
HasProperty	Variable	Representation	MTRepresentation-Type	PropertyType	Optional
HasProperty	Variable	PeriodFilter	Float	PropertyType	Optional
HasMTClassType	Object	<MTDataItemClass>	MTDataItemClassType		Mandatory
HasMTSubClass-Type	Object	<MTDataItemSub-Class>	MTDataItemSubClassType		Optional
HasMT-Composition	Object	<MTComposition>	MTCompositionType		Optional
HasMTSource	Object	<BaseObject>	BaseObjectType		Optional
HasCondition	Event	<MTCondition>	MTConditionType		Optional
HasComponent	Object	Constraints	MTConstraintType		Optional
HasProperty	Variable	SignificantDigits	UInt16	PropertyType	Optional
HasProperty	Variable	Statistic	MTStatisticType	PropertyType	Optional
HasProperty	Variable	NativeUnits	String	PropertyType	Optional
HasProperty	Variable	CoordinateSystem	MTCoordinate-SystemType	PropertyType	Optional
HasProperty	Variable	InitialValue	Double	PropertyType	Optional
HasProperty	Variable	ResetTrigger	MTResetTrigger-Type	PropertyType	Optional
HasProperty	Variable	Units	String	PropertyType	Optional
HasProperty	Variable	MinimumDeltaFilter	Float	PropertyType	Optional
HasProperty	Variable	EngineeringUnits	EUInformation	PropertyType	Mandatory

9.3.15 Defintion of MessageDataType

Table 80: MessageDataType DataType

Field	Type	Optional
NativeCode	String	Optional
Text	String	Mandatory

9.3.16 Defintion of ThreeSpaceSampleDataType

Represents a position in a three space coordinate system. The positions must be given in millimeters.

Table 81: ThreeSpaceSampleDataType DataType

Field	Type	Optional
X	Double	Mandatory
Y	Double	Mandatory
Z	Double	Mandatory

9.3.16.1 Data Type Fields

- X::Double: If not present, must be set to NaN.
- Y::Double: If not present, must be set to NaN.
- Z::Double: If not present, must be set to NaN.

9.4 Conditions

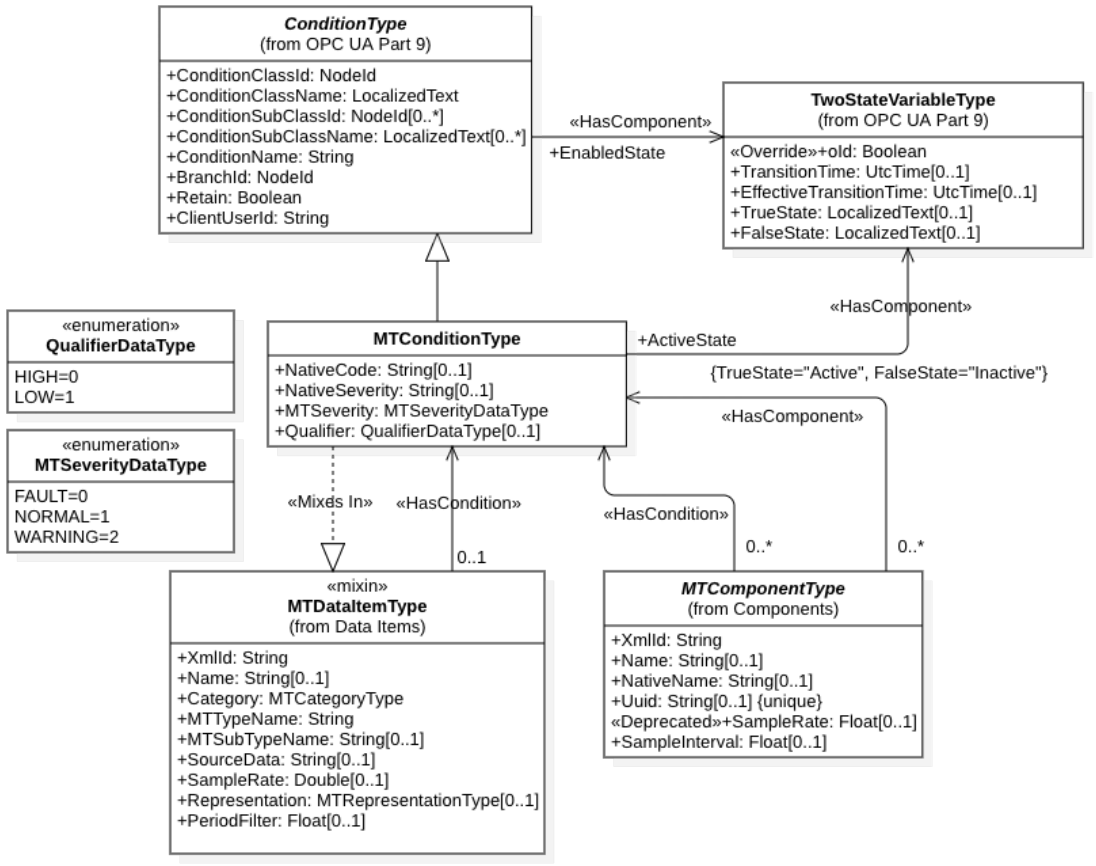

Figure 35: Conditions Diagram

The `Condition` *DataItem* `category` in MTConnect is the mechanism for reporting alarms on the machine classified by type and subType–such as an overload of a motor or high temperature or a collection of system level warnings or faults.

The earlier `Alarm Event` facility changed to a more state-based architecture where the `Condition` was in one of the following three states: Normal – everything is fine, Warning – something is going wrong, but may self-correct, to Fault – a failure that needs manual attention. The change was to address the semantic classification of *Alarms* and remove ambiguity about the `Alarm` severity.

The Condition model is different from the Event model for a lot of the same reasons OPC has the ***ConditionTypes*** as ***Object*** and as an **`Event`**. In UA, the ***Object*** in the address space keeps the state of the condition, whereas the individual **`Alarms`** are discrete events.

In MTConnect, the `Condition` keeps the active state as well. In streaming (*sample request*), the `Conditions` like an event stream. With a *current request*, all active conditions are reported, even if there are multiple active conditions for a given type. Usually, an `Event` can only have one value at a time, MTConnect `Condition` is different and can

have multiple values, as in the case where there are multiple system alarms for a component or a syntax errors in a part program.

The Condition uses the attribute `nativeCode` as you use the BranchId in the UA Condition. Each unique `nativeCode` is consider another activation of the Condition. Only when a `Normal` with no `nativeCode` cleared all active Conditions, or each are cleared separately (going back to a Normal state), does the condition report Normal when for a `current` request.

`Condition` provides a stateful way of managing `Alarms` in a similar way to OPC UA, but since we are not addressing the requirements of an interactive model (we are read-only), we are only reporting on the states of the conditions.

The documentation for the condition behavior in MTConnect can be found in Section 5.7 and 5.8 of [MTConnect Part 3.0] and an overview in [MTConnect Part 2.0].

The MTConnect Data Item with Category of `CONDITION` are mapped to the OPC UA ***ConditionTypes*** in [UA Part 09] with a **`TwoStateVariableType`** that represents the current state of all the active ***branches*** of this `Condition`.

9.4.1 Defintion of `MTConditionType`

The condition type is a derived from the UA `ContitionType`. The Normal state of the condition is indicated by the ActiveState if `FALSE`. The severity is used to represent the MTConnect condition states of Warning and Fault with the values of 500 and 1000 respectively.

An `MTConditionType` instance will be created for event MTConnect *DataItem* with a `category` of `CONDITION`. The `MTConditionType` instance will be instantiated with the **`Retain`** flag set to true when the condition **`ActiveState`** is true.

The **`BrowseName`** of the condition uses the same naming convention as the MTConnect *DataItem* types with `Condition` appended as a suffix. For example the condition with `type` of `TEMPERATURE` will have the browse name of `TemperatureCondition` as opposed to the `MTSampleType` of `Temperature`.

If a `Source` of the *DataItem* is provided, then the **`HasCondition`** relationship will be from the source *DataItem* to the `Condition` instance. Otherwise it will be related to the MTConnect *Component* containing the `MTComponentType` instance.

Table 82: `MTConditionType` Definition

Attribute	Value				
BrowseName	MTConditionType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**Type-Definition**	**Modeling-Rule**
Subtype of ConditionType (See [UA Part 09] Documentation)					
HasProperty	Variable	XmlId	String	PropertyType	Mandatory
HasProperty	Variable	Name	String	PropertyType	Optional
HasProperty	Variable	Category	MTCategoryType	PropertyType	Mandatory
HasProperty	Variable	MTTypeName	String	PropertyType	Mandatory
HasProperty	Variable	MTSubTypeName	String	PropertyType	Optional
HasProperty	Variable	SourceData	String	PropertyType	Optional
HasProperty	Variable	SampleRate	Double	PropertyType	Optional
HasProperty	Variable	Representation	MTRepresentation-Type	PropertyType	Optional
HasProperty	Variable	PeriodFilter	Float	PropertyType	Optional
HasMTClassType	Object	<MTDataItemClass>	MTDataItemClassType		Mandatory
HasMTSubClass-Type	Object	<MTDataItemSub-Class>	MTDataItemSubClassType		Optional
HasMT-Composition	Object	<MTComposition>	MTCompositionType		Optional
HasMTSource	Object	<BaseObject>	BaseObjectType		Optional
HasCondition	Event	<MTCondition>	MTConditionType		Optional
HasComponent	Object	Constraints	MTConstraintType		Optional
HasProperty	Variable	NativeCode	String	PropertyType	Optional
HasProperty	Variable	NativeSeverity	String	PropertyType	Optional
HasProperty	Variable	MTSeverity	MTSeverityData-Type	PropertyType	Mandatory
HasProperty	Variable	Qualifier	QualifierDataType	PropertyType	Optional
HasComponent	Variable	ActiveState	LocalizedText	TwoState-VariableType	Mandatory

9.4.1.1 Referenced Properties and Objects

- `NativeCode::String`: When instantiated in the address space this will represent the `NativeCode` of the last ***Event*** that was received. When the ActiveState becomes False and becomes inactive, then the `NativeCode` will be cleared.

- `NativeSeverity::String`: When instantiated in the address space this will represent the `NativeSeverity` of the last ***Event*** that was received. When the ActiveState becomes False and becomes inactive, then the `NativeSeverity` will be cleared.

- **Allowable Values** for `MTSeverityDataType`

Table 83: `MTSeverityDataType` Enumeration

Name	Index
FAULT	0
NORMAL	1
WARNING	2

- **Allowable Values** for `QualifierDataType`

Table 84: `QualifierDataType` Enumeration

Name	Index
HIGH	0
LOW	1

9.4.1.2 Dependencies and Relationships

- Mixes in `MTDataItemType`, see See section 9.3.6

9.5 Data Item Types

The data item types represent the MTConnect types as defined in Section 8 of the MTConnect Standard Part 2 [MTConnect Part 2.0] and are represented using the XML `type attribute`. The type and sub-type relationships are given in the standard and are not documented in the companion specifiction.

The model of the types is similar to the OPC UA condition classes as given in OPC UA Part 9 [UA Part 09]. Since MTConnect conditions use the same type system as the data items, the representation of the data item type will be derived from the `BaseConditionClassType`. The MTConnect Types will not use the `...ClassType`, but will be the name as specified in the MTConnect standard with `Type` appended.

For example: The `CONTOLLER_MODE` will be given as the `ControllerModeType`. The relationship to the *DataItem* OPC UA types are presented so that it will be easier to map the MTConnect types to the correct super class as given in the OPC UA model.

9.5.1 Defintion of `MTDataItemClassType`

Abstract base class for all the data item class types. The names are created by pascal typing the names and then generating appending `Type`.

Table 85: `MTDataItemClassType` Definition

Attribute	Value				
BrowseName	MTDataItemClassType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of BaseConditionClassType (See [UA Part 09] Documentation)					
HasSubtype	ObjectType	MTSampleClassType		See section 9.6.1	
HasSubtype	ObjectType	MTConditionClassType		See section 9.3.3	
HasSubtype	ObjectType	MTEventClassType		See section 9.3.8	

2160 9.5.2 Defintion of `MTMessageClassType`

Table 86: `MTMessageClassType` Definition

Attribute	Value				
BrowseName	MTMessageClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTEventClassType (See Data Items Documentation)					

2161 9.6 Sample Data Item Types

2162 The MTConnect Standard has the definitions of `Samples` in Section 8.1 of the MTCon-
2163 nect Standard Part 2 [MTConnect Part 2.0] and the definition of the values can be found
2164 in Section 5.3 of MTConnect Standard Part 3 [MTConnect Part 3.0].

2165 The description of each type was copied from the MTConnect Standard, but this not the
2166 definitive text for each type. For authoritative normative text, please refer to the sources
2167 [MTConnect Part 2.0] and [MTConnect Part 3.0].

2168 9.6.1 Defintion of `MTSampleClassType`

2169 The base type class for all data items with a `category` of `SAMPLE`.

Table 87: `MTSampleClassType` Definition

Attribute	Value				
BrowseName	MTSampleClassType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemClassType (See Data Item Types Documentation)					
HasSubtype	ObjectType	MassClassType		See section 9.6.21	
HasSubtype	ObjectType	PathFeedrateClassType		See section 9.6.22	
HasSubtype	ObjectType	PathPositionClassType		See section 9.6.23	
HasSubtype	ObjectType	PHClassType		See section 9.6.24	
HasSubtype	ObjectType	PositionClassType		See section 9.6.25	
HasSubtype	ObjectType	PowerFactorClassType		See section 9.6.26	
HasSubtype	ObjectType	PressureClassType		See section 9.6.27	
HasSubtype	ObjectType	ProcessTimerClassType		See section 9.6.28	
HasSubtype	ObjectType	ResistenceClassType		See section 9.6.29	
HasSubtype	ObjectType	RotaryVelocityClassType		See section 9.6.30	
HasSubtype	ObjectType	SoundLevelClassType		See section 9.6.31	
HasSubtype	ObjectType	StrainClassType		See section 9.6.32	
HasSubtype	ObjectType	TemperatureClassType		See section 9.6.33	
HasSubtype	ObjectType	TensionClassType		See section 9.6.34	
HasSubtype	ObjectType	TiltClassType		See section 9.6.35	
HasSubtype	ObjectType	TorqueClassType		See section 9.6.36	
HasSubtype	ObjectType	VoltAmpereClassType		See section 9.6.37	
HasSubtype	ObjectType	VelocityClassType		See section 9.6.38	
HasSubtype	ObjectType	VoltAmpereReactiveClassType		See section 9.6.39	
HasSubtype	ObjectType	ViscosityClassType		See section 9.6.40	
HasSubtype	ObjectType	VoltageClassType		See section 9.6.41	
HasSubtype	ObjectType	WattageClassType		See section 9.6.42	
Continued...					

References	NodeClass	BrowseName	DataType	TypeDefinition	ModelingRule
HasSubtype	ObjectType	LoadClassType		See section 9.6.2	
HasSubtype	ObjectType	AccelerationClassType		See section 9.6.3	
HasSubtype	ObjectType	AccumulatedTimeClassType		See section 9.6.4	
HasSubtype	ObjectType	AngularAccelerationClassType		See section 9.6.5	
HasSubtype	ObjectType	AngularVelocityClassType		See section 9.6.6	
HasSubtype	ObjectType	AmperageClassType		See section 9.6.7	
HasSubtype	ObjectType	AngleClassType		See section 9.6.8	
HasSubtype	ObjectType	AxisFeedrateClassType		See section 9.6.9	
HasSubtype	ObjectType	ClockTimeClassType		See section 9.6.10	
HasSubtype	ObjectType	ConcentrationClassType		See section 9.6.11	
HasSubtype	ObjectType	ConductivityClassType		See section 9.6.12	
HasSubtype	ObjectType	DisplacementClassType		See section 9.6.13	
HasSubtype	ObjectType	ElectricalEnergyClassType		See section 9.6.14	
HasSubtype	ObjectType	EquipmentTimerClassType		See section 9.6.15	
HasSubtype	ObjectType	FillLevelClassType		See section 9.6.16	
HasSubtype	ObjectType	FlowClassType		See section 9.6.17	
HasSubtype	ObjectType	FrequencyClassType		See section 9.6.18	
HasSubtype	ObjectType	LengthClassType		See section 9.6.19	
HasSubtype	ObjectType	LinearForceClassType		See section 9.6.20	

9.6.2 Defintion of LoadClassType

The measurement of the actual versus the standard rating of a piece of equipment. *PERCENT*

Table 88: LoadClassType Definition

Attribute	Value				
BrowseName	LoadClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

9.6.3 Defintion of AccelerationClassType

Rate of change of velocity. $\frac{MILLIMETER}{SECOND^2}$

Table 89: AccelerationClassType Definition

Attribute	Value				
BrowseName	AccelerationClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

9.6.4 Defintion of AccumulatedTimeClassType

The measurement of accumulated time for an activity or event. *SECOND*

Table 90: AccumulatedTimeClassType Definition

Attribute	Value				
BrowseName	AccumulatedTimeClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

9.6.5 Defintion of AngularAccelerationClassType

Rate of change of angular velocity. $\frac{DEGREE}{SECOND^2}$

Table 91: `AngularAccelerationClassType` Definition

Attribute	Value				
BrowseName	AngularAccelerationClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

2178 9.6.6 Defintion of `AngularVelocityClassType`

2179 Rate of change of angular position. $\frac{DEGREE}{SECOND}$

Table 92: `AngularVelocityClassType` Definition

Attribute	Value				
BrowseName	AngularVelocityClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

2180 9.6.7 Defintion of `AmperageClassType`

2181 The measurement of electrical current. *AMPERE*

Table 93: `AmperageClassType` Definition

Attribute	Value				
BrowseName	AmperageClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

2182 9.6.8 Defintion of `AngleClassType`

2183 The measurement of angular position. *DEGREE*

Table 94: `AngleClassType` Definition

Attribute	Value				
BrowseName	AngleClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

2184 9.6.9 Defintion of `AxisFeedrateClassType`

2185 The feedrate of a linear axis. $\frac{MILLIMETER}{SECOND}$

Table 95: `AxisFeedrateClassType` Definition

Attribute	Value				
BrowseName	AxisFeedrateClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

2186 9.6.10 Defintion of `ClockTimeClassType`

2187 The value provided by a timing device at a specific point in time. *TIMESTAMP*

Table 96: `ClockTimeClassType` Definition

Attribute	Value				
BrowseName	ClockTimeClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

2188 9.6.11 Defintion of `ConcentrationClassType`

2189 Percentage of one component within a mixture of components. *PERCENT*

Table 97: `ConcentrationClassType` Definition

Attribute	Value				
BrowseName	ConcentrationClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

2190 9.6.12 Defintion of `ConductivityClassType`

2191 The ability of a material to conduct electricity. $\frac{SIEMENS}{METER}$

Table 98: `ConductivityClassType` Definition

Attribute	Value				
BrowseName	ConductivityClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

2192 9.6.13 Defintion of `DisplacementClassType`

2193 The change in position of an object. $MILLIMETER$

Table 99: `DisplacementClassType` Definition

Attribute	Value				
BrowseName	DisplacementClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

2194 9.6.14 Defintion of `ElectricalEnergyClassType`

2195 The measurement of electrical energy consumption by a component. $WATT \times SECOND$

Table 100: `ElectricalEnergyClassType` Definition

Attribute	Value				
BrowseName	ElectricalEnergyClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

9.6.15 Defintion of `EquipmentTimerClassType`

The measurement of the amount of time a `SECOND` piece of equipment or a sub-part of a piece of equipment has performed specific activities. Often used to determine when maintenance may be required for the equipment.

Multiple subTypes of `EQUIPMENT_TIMER` MAY be defined. A subType MUST always be specified.

SECOND

Table 101: `EquipmentTimerClassType` Definition

Attribute	Value				
BrowseName	EquipmentTimerClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

9.6.16 Defintion of `FillLevelClassType`

The measurement of the amount of a substance remaining compared to the planned maximum amount of that substance. *PERCENT*

Table 102: `FillLevelClassType` Definition

Attribute	Value				
BrowseName	FillLevelClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

9.6.17 Defintion of `FlowClassType`

The rate of flow of a fluid. $\frac{LITER}{SECOND}$

Table 103: `FlowClassType` Definition

Attribute	Value				
BrowseName	FlowClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

2208 9.6.18 Defintion of `FrequencyClassType`

2209 The measurement of the number of occurrences of a repeating event per unit time. *HERTZ*

Table 104: `FrequencyClassType` Definition

Attribute	Value				
BrowseName	FrequencyClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

2210 9.6.19 Defintion of `LengthClassType`

2211 The length of an object. *MILLIMETER*

Table 105: `LengthClassType` Definition

Attribute	Value				
BrowseName	LengthClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

2212 9.6.19.1 Referenced Properties and Objects

2213 • `Supertype::MTSampleClassType:` The length of an object

2214 9.6.20 Defintion of `LinearForceClassType`

2215 The measure of the push or pull introduced by an actuator or exerted on an object. *NEWTON*

Table 106: `LinearForceClassType` Definition

Attribute	Value				
BrowseName	LinearForceClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

9.6.21 Defintion of `MassClassType`

The measurement of the mass of an object(s) or an amount of material. *KILOGRAM*

Table 107: `MassClassType` Definition

Attribute	Value				
BrowseName	MassClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

9.6.22 Defintion of `PathFeedrateClassType`

The feedrate for the axes, or a single axis, associated with a `Path` component a vector. $\frac{MILLIMETER}{SECOND}$

Table 108: `PathFeedrateClassType` Definition

Attribute	Value				
BrowseName	PathFeedrateClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

9.6.23 Defintion of `PathPositionClassType`

A measured or calculated position of a control point associated with a `Controller` element, or PATH element if provided, of a piece of equipment.

The control point MUST be reported as a set of space-delimited floating-point numbers representing a point in 3-D space. The position of the control point MUST be reported in units of `MILLIMETER` and listed in order of X, Y, and Z referenced to the coordinate system of the piece of equipment.

MILLIMETER($\mathbb{R}^3$)

Table 109: `PathPositionClassType` Definition

Attribute	Value				
BrowseName	PathPositionClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

9.6.24 Defintion of `PHClassType`

The measure of the acidity or alkalinity. *PH*

Table 110: `PHClassType` Definition

Attribute	Value				
BrowseName	PHClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

9.6.25 Defintion of `PositionClassType`

A calculated or measured position related to a Component element.

`POSITION` SHOULD be further defined withacoordinateSytemattribute. If a coordinateSystem attribute is not specified, the position of the control point MUST be reported in `MACHINE` coordinates.

MILLIMETER

Table 111: `PositionClassType` Definition

Attribute	Value				
BrowseName	PositionClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

9.6.26 Defintion of PowerFactorClassType

The measurement of the ratio of real power flowing to a load to the apparent power in that AC circuit. *PERCENT*

Table 112: PowerFactorClassType Definition

Attribute	Value				
BrowseName	PowerFactorClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

9.6.27 Defintion of PressureClassType

The force per unit area exerted by a gas or liquid. *PASCAL*

Table 113: PressureClassType Definition

Attribute	Value				
BrowseName	PressureClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

9.6.28 Defintion of ProcessTimerClassType

The measurement of the amount of time a piece of equipment has performed different types of activities associated with the process being performed at that piece of equipment. Multiple subtypes of PROCESS_TIMER may be defined.

Typically, PROCESS_TIMER SHOULD be modeled as a data item for the Device element, but MAY be modeled for either a Controller or Path Structural Element in the XML document. A subType MUST always be specified.

SECOND

Table 114: `ProcessTimerClassType` Definition

Attribute	Value				
BrowseName	ProcessTimerClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

9.6.29 Defintion of `ResistenceClassType`

The degree to which a substance opposes the passage of an electric current. *OHM*

Table 115: `ResistenceClassType` Definition

Attribute	Value				
BrowseName	ResistenceClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

9.6.30 Defintion of `RotaryVelocityClassType`

The rotational speed of a rotary axis. $\frac{REVOLUTION}{MINUTE}$

Table 116: `RotaryVelocityClassType` Definition

Attribute	Value				
BrowseName	RotaryVelocityClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

9.6.31 Defintion of `SoundLevelClassType`

Measurement of a sound level or sound pressure level relative to atmospheric pressure. *DECIBEL*

Table 117: `SoundLevelClassType` Definition

Attribute	Value				
BrowseName	SoundLevelClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

2257 9.6.32 Defintion of `StrainClassType`

2258 The amount of deformation per unit length of an object when a load is applied. *PERCENT*

Table 118: `StrainClassType` Definition

Attribute	Value				
BrowseName	StrainClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

2259 9.6.33 Defintion of `TemperatureClassType`

2260 The measurement of temperature. *CELSIUS*

Table 119: `TemperatureClassType` Definition

Attribute	Value				
BrowseName	TemperatureClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

2261 9.6.34 Defintion of `TensionClassType`

2262 A measurement of a force that stretches or elongates an object. *NEWTON*

Table 120: `TensionClassType` Definition

Attribute	Value				
BrowseName	TensionClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

2263 9.6.35 Defintion of `TiltClassType`

2264 A measurement of angular displacement. *MICRO* · *RADIAN*

Table 121: `TiltClassType` Definition

Attribute	Value				
BrowseName	TiltClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

2265 9.6.36 Defintion of `TorqueClassType`

2266 The turning force exerted on an object or by an object. $NEWTON \times METER$

Table 122: `TorqueClassType` Definition

Attribute	Value				
BrowseName	TorqueClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

2267 9.6.37 Defintion of `VoltAmpereClassType`

2268 The measure of the apparent power in an electrical circuit, equal to the product of root-
2269 mean-square (RMS) voltage and RMS current (commonly referred to as VA). $VOLT \times$
2270 $AMPERE$

Table 123: `VoltAmpereClassType` Definition

Attribute	Value				
BrowseName	VoltAmpereClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

9.6.38 Defintion of `VelocityClassType`

The rate of change of position. $\frac{MILLIMETER}{SECOND}$

Table 124: `VelocityClassType` Definition

Attribute	Value				
BrowseName	VelocityClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

9.6.39 Defintion of `VoltAmpereReactiveClassType`

The measurement of reactive power in an AC electrical circuit (commonly referred to as VAR). $VOLT \times AMPERE(Reactive)$

Table 125: `VoltAmpereReactiveClassType` Definition

Attribute	Value				
BrowseName	VoltAmpereReactiveClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

9.6.40 Defintion of `ViscosityClassType`

A measurement of a fluid's resistance to flow. $PASCAL \times SECOND$.

Table 126: `ViscosityClassType` Definition

Attribute	Value				
BrowseName	ViscosityClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

9.6.41 Defintion of `VoltageClassType`

The measurement of electrical potential between two points. *VOLT*

Table 127: `VoltageClassType` Definition

Attribute	Value				
BrowseName	VoltageClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

9.6.42 Defintion of `WattageClassType`

The measurement of power flowing through or dissipated by an electrical circuit or piece of equipment. *WATT*

Table 128: `WattageClassType` Definition

Attribute	Value				
BrowseName	WattageClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTSampleClassType (See section 9.6.1)					

9.7 Controlled Vocab Data Item Types

The MTConnect Standard has the definitions of `Events` in Section 8.2 of the MTConnect Standard Part 2 [MTConnect Part 2.0] and the definition of the values can be found in Section 5.5 of MTConnect Standard Part 3 [MTConnect Part 3.0].

The description of each type was copied from the MTConnect Standard, but this not the definitive text for each type. For authoritative normative text, please refer to the sources [MTConnect Part 2.0] and [MTConnect Part 3.0].

9.7.1 Defintion of MTControlledVocabEventClassType

The abstract base type for controlled events that represent states that are provided in related enumerations. These data items will be represented in an object of type MTControlledVocabEventType derived from the OPC UA type **MultiStateValueDiscreteType**

Table 129: MTControlledVocabEventClassType Definition

Attribute	Value				
BrowseName	MTControlledVocabEventClassType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTEventClassType (See Data Items Documentation)					
HasSubtype	ObjectType	DirectionClassType		See section 9.7.13	
HasSubtype	ObjectType	DoorStateClassType		See section 9.7.14	
HasSubtype	ObjectType	EmergencyStopClassType		See section 9.7.15	
HasSubtype	ObjectType	EndOfBarClassType		See section 9.7.16	
HasSubtype	ObjectType	EquipmentModeClassType		See section 9.7.17	
HasSubtype	ObjectType	FunctionalModeClassType		See section 9.7.18	
HasSubtype	ObjectType	SpindleInterlockClassType		See section 9.7.19	
HasSubtype	ObjectType	PathModeClassType		See section 9.7.20	
HasSubtype	ObjectType	PowerStateClassType		See section 9.7.21	
HasSubtype	ObjectType	ProgramEditClassType		See section 9.7.22	
HasSubtype	ObjectType	RotaryModeClassType		See section 9.7.23	
HasSubtype	ObjectType	InterfaceStateClassType		See section 9.7.24	
HasSubtype	ObjectType	MaterialFeedClassType		See section 9.7.25	
HasSubtype	ObjectType	MaterialChangeClassType		See section 9.7.26	
HasSubtype	ObjectType	MaterialRetractClassType		See section 9.7.27	
HasSubtype	ObjectType	MaterialLoadClassType		See section 9.7.28	
HasSubtype	ObjectType	MaterialUnloadClassType		See section 9.7.29	
HasSubtype	ObjectType	OpenDoorClassType		See section 9.7.30	
HasSubtype	ObjectType	CloseDoorClassType		See section 9.7.31	
HasSubtype	ObjectType	OpenChuckClassType		See section 9.7.32	
HasSubtype	ObjectType	CloseChuckClassType		See section 9.7.33	
HasSubtype	ObjectType	PartChangeClassType		See section 9.7.34	
Continued...					

References	NodeClass	BrowseName	DataType	TypeDefinition	ModelingRule
HasSubtype	ObjectType	ActuatorStateClassType		See section 9.7.2	
HasSubtype	ObjectType	AvailabilityClassType		See section 9.7.3	
HasSubtype	ObjectType	AxisCouplingClassType		See section 9.7.4	
HasSubtype	ObjectType	AxisInterlockClassType		See section 9.7.5	
HasSubtype	ObjectType	AxisStateClassType		See section 9.7.6	
HasSubtype	ObjectType	ChuckInterlockClassType		See section 9.7.7	
HasSubtype	ObjectType	ChuckStateClassType		See section 9.7.8	
HasSubtype	ObjectType	ControllerModeClassType		See section 9.7.9	
HasSubtype	ObjectType	ExecutionClassType		See section 9.7.10	
HasSubtype	ObjectType	CompositionStateClassType		See section 9.7.11	
HasSubtype	ObjectType	ControllerModeOverrideClassType		See section 9.7.12	

9.7.2 Defintion of ActuatorStateClassType

Represents the operational state of an apparatus for moving or controlling.

Table 130: ActuatorStateClassType Definition

Attribute	Value				
BrowseName	ActuatorStateClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	ActiveStateDataType	ActiveStateDataType	Mandatory

9.7.2.1 Referenced Properties and Objects

- **Allowable Values** for ActiveStateDataType

Table 131: ActiveStateDataType Enumeration

Name	Index
ACTIVE	0
INACTIVE	1

9.7.3 Defintion of AvailabilityClassType

Represents the Agent's ability to communicate with the data source. This MUST be provided for a Device Element and MAY be provided for any other Structural Element.

Table 132: `AvailabilityClassType` Definition

Attribute	Value				
BrowseName	AvailabilityClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	AvailabilityDataType	AvailabilityDataType	Mandatory

9.7.3.1 Referenced Properties and Objects

- **Allowable Values** for `AvailabilityDataType`

Table 133: `AvailabilityDataType` Enumeration

Name	Index
AVAILABLE	0
UNAVAILABLE	1

9.7.4 Defintion of `AxisCouplingClassType`

Describes the way the axes will be associated to each other. This is used in conjunction with `COUPLED_AXES` to indicate the way they are interacting.

The coupling MUST be viewed from the perspective of a specific axis. Therefore, a `MASTER` coupling indicates that this axis is the master for the `COUPLED_AXES`.

Table 134: `AxisCouplingClassType` Definition

Attribute	Value				
BrowseName	AxisCouplingClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	AxisCouplingDataType	AxisCouplingDataType	Mandatory

9.7.4.1 Referenced Properties and Objects

- **Allowable Values** for `AxisCouplingDataType`

Table 135: `AxisCouplingDataType` Enumeration

Name	Index
MASTER	0
SLAVE	1
SYNCHRONOUS	2
TANDEM	3

2311 9.7.5 Defintion of `AxisInterlockClassType`

2312 An indicator of the state of the axis lockout function when power has been removed and
2313 the axis is allowed to move freely.

Table 136: `AxisInterlockClassType` Definition

Attribute	Value				
BrowseName	AxisInterlockClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	ActiveStateDataType	ActiveStateDataType	Mandatory

2314 9.7.5.1 Referenced Properties and Objects

2315 • **Allowable Values** for `ActiveStateDataType`

Table 137: `ActiveStateDataType` Enumeration

Name	Index
ACTIVE	0
INACTIVE	1

2316 9.7.6 Defintion of `AxisStateClassType`

2317 An indicator of the controlled state of a `LINEAR` or `ROTARY` component representing an
2318 axis.

Table 138: `AxisStateClassType` Definition

Attribute	Value				
BrowseName	AxisStateClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	AxisStateDataType	AxisStateDataType	Mandatory

2319 9.7.6.1 Referenced Properties and Objects

2320 • **Allowable Values** for `AxisStateDataType`

Table 139: `AxisStateDataType` Enumeration

Name	Index
HOME	0
PARKED	1
STOPPED	2
TRAVEL	3

2321 9.7.7 Defintion of `ChuckInterlockClassType`

2322 An indication of the state of an interlock function or control logic state intended to prevent
2323 the associated `Chuck` composition or function from being operated.

Table 140: `ChuckInterlockClassType` Definition

Attribute	Value				
BrowseName	ChuckInterlockClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	ActiveStateDataType	ActiveStateDataType	Mandatory

2324 9.7.7.1 Referenced Properties and Objects

2325 • **Allowable Values** for `ActiveStateDataType`

Table 141: `ActiveStateDataType` Enumeration

Name	Index
ACTIVE	0
INACTIVE	1

9.7.8 Defintion of `ChuckStateClassType`

An indication of the operating state of a mechanism that holds a part or stock material during a manufacturing process. It may also represent a mechanism that holds any other mechanism in place within a piece of equipment.

Table 142: `ChuckStateClassType` Definition

Attribute	Value				
BrowseName	ChuckStateClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	OpenStateDataType	OpenStateDataType	Mandatory

9.7.8.1 Referenced Properties and Objects

- **Allowable Values** for `OpenStateDataType`

Table 143: `OpenStateDataType` Enumeration

Name	Index
CLOSED	0
OPEN	1
UNLATCHED	2

9.7.9 Defintion of `ControllerModeClassType`

The current mode of the `Controller` component.

Table 144: `ControllerModeClassType` Definition

Attribute	Value				
BrowseName	ControllerModeClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modeling-Rule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	ControllerModeDataType	ControllerMode-DataType	Mandatory

9.7.9.1 Referenced Properties and Objects

- **Allowable Values** for `ControllerModeDataType`

Table 145: `ControllerModeDataType` Enumeration

Name	Index
AUTOMATIC	0
EDIT	1
MANUAL	2
MANUAL_DATA_INPUT	3
SEMI_AUTOMATIC	4

9.7.10 Defintion of `ExecutionClassType`

The execution status of the `Controller` or `Path`.

Table 146: `ExecutionClassType` Definition

Attribute	Value				
BrowseName	ExecutionClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	ExecutionDataType	ExecutionDataType	Mandatory

9.7.10.1 Referenced Properties and Objects

- **Allowable Values** for `ExecutionDataType`

Table 147: `ExecutionDataType` Enumeration

Name	Index
ACTIVE	0
FEED_HOLD	1
INTERRUPTED	2
OPTIONAL_STOP	3
READY	4
PROGRAM_COMPLETED	5
PROGRAM_STOPPED	6
STOPPED	7

9.7.11 Defintion of `CompositionStateClassType`

An indication of the operating condition of a mechanism represented by a `Composition` type element.

A `subType` MUST always be specified.

A `compositionId` MUST always be specified.

Table 148: `CompositionStateClassType` Definition

Attribute	Value				
BrowseName	CompositionStateClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modeling-Rule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	CompositionStateDataType	CompositionState-DataType	Mandatory

9.7.11.1 Referenced Properties and Objects

- **Allowable Values** for `CompositionStateDataType`

Table 149: `CompositionStateDataType` Enumeration

Name	Index
ACTIVE	0
CLOSED	1
DOWN	2
INACTIVE	3
LEFT	4
OFF	5
ON	6
OPEN	7
RIGHT	8
TRANSITIONING	9
UNLATCHED	10
UP	11

2347 9.7.12 Defintion of `ControllerModeOverrideClassType`

2348 A setting or operator selection that changes the behavior of a piece of equipment.

Table 150: `ControllerModeOverrideClassType` Definition

Attribute	Value				
BrowseName	ControllerModeOverrideClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	OnOffDataType	OnOffDataType	Mandatory

2349 9.7.12.1 Referenced Properties and Objects

2350 • **Allowable Values** for `OnOffDataType`

Table 151: `OnOffDataType` Enumeration

Name	Index
OFF	0
ON	1

2351 9.7.13 Defintion of DirectionClassType

2352 The direction of motion. A subType MUST always be specified.

Table 152: DirectionClassType Definition

Attribute	Value				
BrowseName	DirectionClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	DirectionDataType	DirectionDataType	Mandatory

2353 9.7.13.1 Referenced Properties and Objects

2354 • **Allowable Values** for DirectionDataType

Table 153: DirectionDataType Enumeration

Name	Index
CLOCKWISE	0
COUNTER_CLOCKWISE	1
NEGATIVE	2
POSITIVE	3

2355 9.7.14 Defintion of DoorStateClassType

2356 The opened or closed state of the door.

Table 154: DoorStateClassType Definition

Attribute	Value				
BrowseName	DoorStateClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	OpenStateDataType	OpenStateDataType	Mandatory

2357 9.7.14.1 Referenced Properties and Objects

2358 • **Allowable Values** for OpenStateDataType

Table 155: `OpenStateDataType` Enumeration

Name	Index
CLOSED	0
OPEN	1
UNLATCHED	2

2359 9.7.15 Defintion of `EmergencyStopClassType`

2360 The current state of the emergency stop signal.

Table 156: `EmergencyStopClassType` Definition

Attribute	Value				
BrowseName	EmergencyStopClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modeling-Rule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	EmergencyStopDataType	EmergencyStop-DataType	Mandatory

2361 9.7.15.1 Referenced Properties and Objects

2362 • **Allowable Values** for `EmergencyStopDataType`

Table 157: `EmergencyStopDataType` Enumeration

Name	Index
ARMED	0
TRIGGERED	1

2363 9.7.16 Defintion of `EndOfBarClassType`

2364 An indication of whether the end of a piece of bar stock being feed by a bar feeder has
2365 been reached.

Table 158: `EndOfBarClassType` Definition

Attribute	Value				
BrowseName	EndOfBarClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	YesNoDataType	YesNoDataType	Mandatory

9.7.16.1 Referenced Properties and Objects

- **Allowable Values** for `YesNoDataType`

Table 159: `YesNoDataType` Enumeration

Name	Index
NO	0
YES	1

9.7.17 Defintion of `EquipmentModeClassType`

An indication that a piece of equipment, or a sub-part of a piece of equipment, is performing specific types of activities.

A `subType` MUST always be specified.

Table 160: `EquipmentModeClassType` Definition

Attribute	Value				
BrowseName	EquipmentModeClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	OnOffDataType	OnOffDataType	Mandatory

9.7.17.1 Referenced Properties and Objects

- **Allowable Values** for `OnOffDataType`

Table 161: `OnOffDataType` Enumeration

Name	Index
OFF	0
ON	1

2374 9.7.18 Defintion of `FunctionalModeClassType`

2375 The current intended production status of the device or component.

2376 Typically, the `FUNCTIONAL_MODE` SHOULD be modeled as a data item for the Device
2377 element, but MAY be modeled for any Structural Element in the XML document.

Table 162: `FunctionalModeClassType` Definition

Attribute	Value				
BrowseName	FunctionalModeClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modeling-Rule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	FunctionalModeDataType	FunctionalMode-DataType	Mandatory

2378 9.7.18.1 Referenced Properties and Objects

2379 • **Allowable Values** for `FunctionalModeDataType`

Table 163: `FunctionalModeDataType` Enumeration

Name	Index
MAINTENANCE	0
PRODUCTION	1
PROCESS_DEVELOPMENT	2
SETUP	3
TEARDOWN	4

2380 9.7.19 Defintion of `SpindleInterlockClassType`

2381 An indication of the status of the spindle for a piece of equipment when power has been
2382 removed and it is free to rotate.

Table 164: `SpindleInterlockClassType` Definition

Attribute	Value				
BrowseName	SpindleInterlockClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	ActiveStateDataType	ActiveStateDataType	Mandatory

2383 9.7.19.1 Referenced Properties and Objects

2384 • **Allowable Values** for `ActiveStateDataType`

Table 165: `ActiveStateDataType` Enumeration

Name	Index
ACTIVE	0
INACTIVE	1

2385 9.7.20 Defintion of `PathModeClassType`

2386 Describes the operational relationship between a `PATH` *Structural Element* and another
2387 `PATH` *Structural Element* for pieces of equipment comprised of multiple logical groupings
2388 of controlled axes or other logical operations.

Table 166: `PathModeClassType` Definition

Attribute	Value				
BrowseName	PathModeClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	PathModeDataType	PathModeDataType	Mandatory

2389 9.7.20.1 Referenced Properties and Objects

2390 • **Allowable Values** for `PathModeDataType`

Table 167: `PathModeDataType` Enumeration

Name	Index
INDEPENDENT	0
MASTER	1
MIRROR	2
SYNCHRONOUS	3

2391 9.7.21 Defintion of `PowerStateClassType`

2392 The indication of the status of the source of energy for a *Structural Element* to allow it to
2393 perform its intended function or the state of an enabling signal providing permission for
2394 the *Structural Element* to perform its functions.

2395 DEPRECATION WARNING: `PowerState` may be deprecated in the future.

Table 168: `PowerStateClassType` Definition

Attribute	Value				
BrowseName	PowerStateClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	OnOffDataType	OnOffDataType	Mandatory

2396 9.7.21.1 Referenced Properties and Objects

2397 • **Allowable Values** for `OnOffDataType`

Table 169: `OnOffDataType` Enumeration

Name	Index
OFF	0
ON	1

2398 9.7.22 Defintion of `ProgramEditClassType`

2399 An indication of the status of the `Controller` component's program editing mode.

2400 On many controls, a program can be edited while another program is currently being
2401 executed.

Table 170: `ProgramEditClassType` Definition

Attribute	Value				
BrowseName	ProgramEditClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	ProgramEditDataType	ProgramEditDataType	Mandatory

2402 9.7.22.1 Referenced Properties and Objects

2403 • **Allowable Values** for `ProgramEditDataType`

Table 171: `ProgramEditDataType` Enumeration

Name	Index
ACTIVE	0
NOT_READY	1
READY	2

2404 9.7.23 Defintion of `RotaryModeClassType`

2405 The current operating mode for a `Rotary` type axis.

Table 172: `RotaryModeClassType` Definition

Attribute	Value				
BrowseName	RotaryModeClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	RotaryModeDataType	RotaryModeDataType	Mandatory

2406 9.7.23.1 Referenced Properties and Objects

2407 • **Allowable Values** for `RotaryModeDataType`

Table 173: `RotaryModeDataType` Enumeration

Name	Index
CONTOUR	0
INDEX	1
SPINDLE	2

2408 9.7.24 Defintion of `InterfaceStateClassType`

2409 The current functional or operational state of an Interface type element indicating whether
2410 the Interface is active or not currently functioning.

Table 174: `InterfaceStateClassType` Definition

Attribute	Value				
BrowseName	InterfaceStateClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Modeling-Rule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	InterfaceStatusDataType	InterfaceStatusData-Type	Mandatory

2411 9.7.24.1 Referenced Properties and Objects

2412 • **Allowable Values** for `InterfaceStatusDataType`

Table 175: `InterfaceStatusDataType` Enumeration

Name	Index
DISABLED	0
ENABLED	1

2413 9.7.25 Defintion of `MaterialFeedClassType`

2414 Service to advance material or feed product to a piece of equipment from a continuous or
2415 bulk source.

Table 176: `MaterialFeedClassType` Definition

Attribute	Value				
BrowseName	MaterialFeedClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	InterfaceStateDataType	InterfaceStateData-Type	Mandatory

2416 9.7.25.1 Referenced Properties and Objects

2417 • **Allowable Values** for `InterfaceStateDataType`

Table 177: `InterfaceStateDataType` Enumeration

Name	Index
ACTIVE	0
COMPLETE	1
FAIL	2
NOT_READY	4
READY	5

2418 9.7.26 Defintion of `MaterialChangeClassType`

2419 Service to change the type of material or product being loaded or fed to a piece of equip-
2420 ment.

Table 178: `MaterialChangeClassType` Definition

Attribute	Value				
BrowseName	MaterialChangeClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	InterfaceStateDataType	InterfaceStateData-Type	Mandatory

2421 9.7.26.1 Referenced Properties and Objects

2422 • **Allowable Values** for `InterfaceStateDataType`

Table 179: `InterfaceStateDataType` Enumeration

Name	Index
ACTIVE	0
COMPLETE	1
FAIL	2
NOT_READY	4
READY	5

9.7.27 Defintion of `MaterialRetractClassType`

Service to remove or retract material or product.

Table 180: `MaterialRetractClassType` Definition

Attribute	Value				
BrowseName	MaterialRetractClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	InterfaceStateDataType	InterfaceStateData-Type	Mandatory

9.7.27.1 Referenced Properties and Objects

- **Allowable Values** for `InterfaceStateDataType`

Table 181: `InterfaceStateDataType` Enumeration

Name	Index
ACTIVE	0
COMPLETE	1
FAIL	2
NOT_READY	4
READY	5

9.7.28 Defintion of `MaterialLoadClassType`

Service to load a piece of material or product.

Table 182: `MaterialLoadClassType` Definition

Attribute	Value				
BrowseName	MaterialLoadClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	InterfaceStateDataType	InterfaceStateData-Type	Mandatory

2429 9.7.28.1 Referenced Properties and Objects

2430 • **Allowable Values** for `InterfaceStateDataType`

Table 183: `InterfaceStateDataType` Enumeration

Name	Index
ACTIVE	0
COMPLETE	1
FAIL	2
NOT_READY	4
READY	5

2431 9.7.29 Defintion of `MaterialUnloadClassType`

2432 Service to unload a piece of material or product.

Table 184: `MaterialUnloadClassType` Definition

Attribute	Value				
BrowseName	MaterialUnloadClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	InterfaceStateDataType	InterfaceStateData-Type	Mandatory

2433 9.7.29.1 Referenced Properties and Objects

2434 • **Allowable Values** for `InterfaceStateDataType`

Table 185: `InterfaceStateDataType` Enumeration

Name	Index
ACTIVE	0
COMPLETE	1
FAIL	2
NOT_READY	4
READY	5

2435 9.7.30 Defintion of `OpenDoorClassType`

2436 Service to open a door.

Table 186: `OpenDoorClassType` Definition

Attribute	Value				
BrowseName	OpenDoorClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	InterfaceStateDataType	InterfaceStateData-Type	Mandatory

2437 9.7.30.1 Referenced Properties and Objects

2438 • **Allowable Values** for `InterfaceStateDataType`

Table 187: `InterfaceStateDataType` Enumeration

Name	Index
ACTIVE	0
COMPLETE	1
FAIL	2
NOT_READY	4
READY	5

2439 9.7.31 Defintion of `CloseDoorClassType`

2440 Service to close a door.

Table 188: `CloseDoorClassType` Definition

Attribute	Value				
BrowseName	CloseDoorClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	InterfaceStateDataType	InterfaceStateData-Type	Mandatory

2441 9.7.31.1 Referenced Properties and Objects

2442 • **Allowable Values** for `InterfaceStateDataType`

Table 189: `InterfaceStateDataType` Enumeration

Name	Index
ACTIVE	0
COMPLETE	1
FAIL	2
NOT_READY	4
READY	5

2443 9.7.32 Defintion of `OpenChuckClassType`

2444 Service to open a chuck.

Table 190: `OpenChuckClassType` Definition

Attribute	Value				
BrowseName	OpenChuckClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	InterfaceStateDataType	InterfaceStateData-Type	Mandatory

2445 9.7.32.1 Referenced Properties and Objects

2446 • **Allowable Values** for `InterfaceStateDataType`

Table 191: `InterfaceStateDataType` Enumeration

Name	Index
ACTIVE	0
COMPLETE	1
FAIL	2
NOT_READY	4
READY	5

9.7.33 Defintion of `CloseChuckClassType`

Service to close a chuck.

Table 192: `CloseChuckClassType` Definition

Attribute	Value				
BrowseName	CloseChuckClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	InterfaceStateDataType	InterfaceStateData-Type	Mandatory

9.7.33.1 Referenced Properties and Objects

- **Allowable Values** for `InterfaceStateDataType`

Table 193: `InterfaceStateDataType` Enumeration

Name	Index
ACTIVE	0
COMPLETE	1
FAIL	2
NOT_READY	4
READY	5

9.7.34 Defintion of `PartChangeClassType`

Service to change the part or product associated with a piece of equipment to a different part or product.

Table 194: `PartChangeClassType` Definition

Attribute	Value				
BrowseName	PartChangeClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTControlledVocabEventClassType (See section 9.7.1)					
HasProperty	Variable	EnumValues	InterfaceStateDataType	InterfaceStateData-Type	Mandatory

9.7.34.1 Referenced Properties and Objects

- **Allowable Values** for `InterfaceStateDataType`

Table 195: `InterfaceStateDataType` Enumeration

Name	Index
ACTIVE	0
COMPLETE	1
FAIL	2
NOT_READY	4
READY	5

9.8 Numeric Event Data Item Types

The MTConnect Standard has the definitions of `Events` in Section 8.2 of the MTConnect Standard Part 2 [MTConnect Part 2.0] and the definition of the values can be found in Section 5.5 of MTConnect Standard Part 3 [MTConnect Part 3.0].

The description of each type was copied from the MTConnect Standard, but this not the definitive text for each type. For authoritative normative text, please refer to the sources [MTConnect Part 2.0] and [MTConnect Part 3.0].

9.8.1 Defintion of `MTNumericEventClassType`

The root type for all of the event types that have numeric `CDATA`.

Table 196: `MTNumericEventClassType` Definition

Attribute	Value				
BrowseName	MTNumericEventClassType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTEventClassType (See Data Items Documentation)					
HasSubtype	ObjectType	AxisFeedrateOverrideClassType		See section 9.8.2	
HasSubtype	ObjectType	BlockCountClassType		See section 9.8.3	
HasSubtype	ObjectType	HardnessClassType		See section 9.8.4	
HasSubtype	ObjectType	LineNumberClassType		See section 9.8.5	
HasSubtype	ObjectType	PartCountClassType		See section 9.8.6	
HasSubtype	ObjectType	RotaryVelocityOverrideClassType		See section 9.8.7	

9.8.2 Defintion of `AxisFeedrateOverrideClassType`

The value of a signal or calculation issued to adjust the feedrate of an individual linear type axis.

The value provided for `AXIS_FEEDRATE_OVERRIDE` is expressed as a percentage of the designated feedrate for the axis.

When `AXIS_FEEDRATE_OVERRIDE` is applied, the resulting commanded feedrate for the axis is limited to the value of the original feedrate multiplied by the value of the `AXIS_FEEDRATE_OVERRIDE`.

There MAY be different subtypes of `AXIS_FEEDRATE_OVERRIDE`; each representing an override value for a designated subtype of feedrate depending on the state of operation of the axis. The subtypes of operation of an axis are currently defined as `PROGRAMMED`, `JOG`, and `RAPID`.

Table 197: `AxisFeedrateOverrideClassType` Definition

Attribute	Value				
BrowseName	AxisFeedrateOverrideClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTNumericEventClassType (See section 9.8.1)					

9.8.3 Defintion of `BlockCountClassType`

The total count of the number of blocks of program code that have been executed since execution started.

BLOCK_COUNT counts blocks of program code executed regardless of program structure (e.g., looping or branching within the program).

The starting value for BLOCK_COUNT MAY be established by an initial value provided in the Constraint element defined for the data item.

Table 198: BlockCountClassType Definition

Attribute	Value				
BrowseName	BlockCountClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTNumericEventClassType (See section 9.8.1)					

9.8.4 Defintion of HardnessClassType

The measurement of the hardness of a material.

The measurement does not provide a unit.

A subType MUST always be specified to designate the hardness scale associated with the measurement.

Table 199: HardnessClassType Definition

Attribute	Value				
BrowseName	HardnessClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTNumericEventClassType (See section 9.8.1)					

9.8.5 Defintion of LineNumberClassType

A reference to the position of a block of program code within a control program. The line number MAY represent either an absolute position starting with the first line of the program or an incremental position relative to the occurrence of the last LINE_LABEL. LINE_NUMBER does not change subject to any looping or branching in a control program.

A subType MUST be defined.

Table 200: `LineNumberClassType` Definition

Attribute	Value				
BrowseName	LineNumberClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTNumericEventClassType (See section 9.8.1)					

9.8.6 Defintion of `PartCountClassType`

The current count of parts produced as represented by the Controller. The valid data value MUST be an integer value.

Table 201: `PartCountClassType` Definition

Attribute	Value				
BrowseName	PartCountClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTNumericEventClassType (See section 9.8.1)					

9.8.7 Defintion of `RotaryVelocityOverrideClassType`

A command issued to adjust the programmed velocity for a Rotary type axis.

This command represents a percentage change to the velocity calculated by a logic or motion program or set by a switch for a Rotary type axis. `ROTARY_VELOCITY_OVERRIDE` is expressed as a percentage of the programmed `ROTARY_VELOCITY`.

Table 202: `RotaryVelocityOverrideClassType` Definition

Attribute	Value				
BrowseName	RotaryVelocityOverrideClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTNumericEventClassType (See section 9.8.1)					

9.9 String Event Data Item Types

The MTConnect Standard has the definitions of `Events` in Section 8.2 of the MTConnect Standard Part 2 [MTConnect Part 2.0] and the definition of the values can be found in Section 5.5 of MTConnect Standard Part 3 [MTConnect Part 3.0].

The description of each type was copied from the MTConnect Standard, but this not the definitive text for each type. For authoritative normative text, please refer to the sources [MTConnect Part 2.0] and [MTConnect Part 3.0].

9.9.1 Defintion of `MTStringEventClassType`

The base UA ***Type*** for all *DataItems* that have a non-specific text representation.

Table 203: `MTStringEventClassType` Definition

Attribute	**Value**				
BrowseName	MTStringEventClassType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTEventClassType (See Data Items Documentation)					
HasSubtype	ObjectType	LineLabelClassType		See section 9.9.4	
HasSubtype	ObjectType	MaterialClassType		See section 9.9.5	
HasSubtype	ObjectType	OperatorIdClassType		See section 9.9.6	
HasSubtype	ObjectType	PalletIdClassType		See section 9.9.7	
HasSubtype	ObjectType	PartIdClassType		See section 9.9.8	
HasSubtype	ObjectType	PartNumberClassType		See section 9.9.9	
HasSubtype	ObjectType	ProgramClassType		See section 9.9.10	
HasSubtype	ObjectType	ProgramEditNameClassType		See section 9.9.11	
HasSubtype	ObjectType	ProgramHeaderClassType		See section 9.9.12	
HasSubtype	ObjectType	ProgramCommentClassType		See section 9.9.13	
HasSubtype	ObjectType	SerialNumberClassType		See section 9.9.14	
HasSubtype	ObjectType	ToolAssetIdClassType		See section 9.9.15	
HasSubtype	ObjectType	ToolNumberClassType		See section 9.9.16	
HasSubtype	ObjectType	ToolOffsetClassType		See section 9.9.17	
HasSubtype	ObjectType	UserClassType		See section 9.9.18	
HasSubtype	ObjectType	WireClassType		See section 9.9.19	
HasSubtype	ObjectType	WorkholdingClassType		See section 9.9.20	
HasSubtype	ObjectType	WorkOffsetClassType		See section 9.9.21	
HasSubtype	ObjectType	MessageClassType		See section 9.9.22	
HasSubtype	ObjectType	AssetChangedClassType		See section 9.9.23	
HasSubtype	ObjectType	AssetRemovedClassType		See section 9.9.24	
HasSubtype	ObjectType	LineClassType		See section 9.9.25	
Continued...					

References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
HasSubtype	ObjectType	BlockClassType		See section 9.9.2	
HasSubtype	ObjectType	CoupledAxesClassType		See section 9.9.3	

9.9.2 Defintion of BlockClassType

The line of code or command being executed by a Controller *Structural Element*.

The value reported for Block MUST include the entire expression for a line of program code, including all parameters.

Table 204: BlockClassType Definition

Attribute	Value				
BrowseName	BlockClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTStringEventClassType (See section 9.9.1)					

9.9.2.1 Referenced Properties and Objects

- Supertype::MTStringEventClassType: The line of code or command being executed by a Controller *Structural Element*.

9.9.3 Defintion of CoupledAxesClassType

Refers to the set of associated axes.

The valid data value for COUPLED_AXES SHOULD be a space-delimited set of axes reported as the value of the name attribute for each axis. If name is not available, the piece of equipment MUST report the value of the nativeName attribute for each axis.

Table 205: CoupledAxesClassType Definition

Attribute	Value				
BrowseName	CoupledAxesClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTStringEventClassType (See section 9.9.1)					

9.9.4 Defintion of LineLabelClassType

An optional identifier for a BLOCK of code in a PROGRAM.

Table 206: `LineLabelClassType` Definition

Attribute	Value				
BrowseName	LineLabelClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTStringEventClassType (See section 9.9.1)					

2526 9.9.5 Defintion of `MaterialClassType`

2527 The identifier of a material used or consumed in the manufacturing process.

Table 207: `MaterialClassType` Definition

Attribute	Value				
BrowseName	MaterialClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTStringEventClassType (See section 9.9.1)					

2528 9.9.6 Defintion of `OperatorIdClassType`

2529 The identifier of the person currently responsible for operating the piece of equipment.

2530 DEPRECATION WARNING: May be deprecated in the future. See `USER` below.

Table 208: `OperatorIdClassType` Definition

Attribute	Value				
BrowseName	OperatorIdClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTStringEventClassType (See section 9.9.1)					

2531 9.9.7 Defintion of `PalletIdClassType`

2532 The identifier for a pallet.

Table 209: `PalletIdClassType` Definition

Attribute	Value				
BrowseName	PalletIdClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTStringEventClassType (See section 9.9.1)					

9.9.8 Defintion of `PartIdClassType`

An identifier of a part in a manufacturing operation.

Table 210: `PartIdClassType` Definition

Attribute	Value				
BrowseName	PartIdClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTStringEventClassType (See section 9.9.1)					

9.9.9 Defintion of `PartNumberClassType`

An identifier of a part or product moving through the manufacturing process.

Table 211: `PartNumberClassType` Definition

Attribute	Value				
BrowseName	PartNumberClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTStringEventClassType (See section 9.9.1)					

9.9.10 Defintion of `ProgramClassType`

The name of the logic or motion program being executed by the `Controller` or `Path` component.

Table 212: `ProgramClassType` Definition

Attribute	Value				
BrowseName	ProgramClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTStringEventClassType (See section 9.9.1)					

2540 9.9.11 Defintion of `ProgramEditNameClassType`

2541 The name of the program being edited. This is used in conjunction with `PROGRAM_EDIT`
2542 when in `ACTIVE` state.

Table 213: `ProgramEditNameClassType` Definition

Attribute	Value				
BrowseName	ProgramEditNameClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTStringEventClassType (See section 9.9.1)					

2543 9.9.12 Defintion of `ProgramHeaderClassType`

2544 The non-executable header section of the control program.

Table 214: `ProgramHeaderClassType` Definition

Attribute	Value				
BrowseName	ProgramHeaderClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTStringEventClassType (See section 9.9.1)					

2545 9.9.13 Defintion of `ProgramCommentClassType`

2546 A comment or non-executable statement in the control program.

Table 215: `ProgramCommentClassType` Definition

Attribute	Value				
BrowseName	ProgramCommentClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTStringEventClassType (See section 9.9.1)					

2547 9.9.14 Defintion of SerialNumberClassType

2548 The serial number associated with a `Component`, `Asset`, or `Device`.

Table 216: `SerialNumberClassType` Definition

Attribute	Value				
BrowseName	SerialNumberClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTStringEventClassType (See section 9.9.1)					

2549 9.9.15 Defintion of ToolAssetIdClassType

2550 The identifier of an individual tool asset

Table 217: `ToolAssetIdClassType` Definition

Attribute	Value				
BrowseName	ToolAssetIdClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTStringEventClassType (See section 9.9.1)					

2551 9.9.16 Defintion of ToolNumberClassType

2552 The identifier of a tool provided by the piece of equipment controller.

Table 218: `ToolNumberClassType` Definition

Attribute	Value				
BrowseName	ToolNumberClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTStringEventClassType (See section 9.9.1)					

9.9.17 Defintion of `ToolOffsetClassType`

A reference to the tool offset variables applied to the active cutting tool associated with a Path in a Controller type component.

The valid data value MUST be a text string.

The reported value returned for `TOOL_OFFSET` identifies the location in a table or list where the actual tool offset values are stored.

A `subType` MUST always be specified.

Table 219: `ToolOffsetClassType` Definition

Attribute	Value				
BrowseName	ToolOffsetClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTStringEventClassType (See section 9.9.1)					

9.9.18 Defintion of `UserClassType`

The identifier of the person currently responsible for operating the piece of equipment.

A `subType` MUST always be specified.

Table 220: `UserClassType` Definition

Attribute	Value				
BrowseName	UserClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTStringEventClassType (See section 9.9.1)					

9.9.19 Defintion of WireClassType

The identifier for the type of wire used as the cutting mechanism in Electrical Discharge Machining or similar processes.

Table 221: WireClassType Definition

Attribute	Value				
BrowseName	WireClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTStringEventClassType (See section 9.9.1)					

9.9.20 Defintion of WorkholdingClassType

The identifier for the workholding currently in use.

Table 222: WorkholdingClassType Definition

Attribute	Value				
BrowseName	WorkholdingClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTStringEventClassType (See section 9.9.1)					

9.9.21 Defintion of WorkOffsetClassType

A reference to the offset variables for a work piece or part associated with a Path in a Controller type component.

The valid data value MUST be a text string.

The reported value returned for WORK_OFFSET identifies the location in a table or list where the actual tool offset values are stored.

Table 223: WorkOffsetClassType Definition

Attribute	Value				
BrowseName	WorkOffsetClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTStringEventClassType (See section 9.9.1)					

9.9.22 Defintion of MessageClassType

Table 224: MessageClassType Definition

Attribute	Value				
BrowseName	MessageClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTStringEventClassType (See section 9.9.1)					

9.9.23 Defintion of AssetChangedClassType

Table 225: AssetChangedClassType Definition

Attribute	Value				
BrowseName	AssetChangedClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTStringEventClassType (See section 9.9.1)					

9.9.24 Defintion of AssetRemovedClassType

Table 226: AssetRemovedClassType Definition

Attribute	Value				
BrowseName	AssetRemovedClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTStringEventClassType (See section 9.9.1)					

9.9.25 Defintion of LineClassType

Table 227: `LineClassType` Definition

Attribute	Value				
BrowseName	LineClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTStringEventClassType (See section 9.9.1)					

9.10 Condition Data Item Types

9.10.1 Defintion of `ActuatorClassType`

Table 228: `ActuatorClassType` Definition

Attribute	Value				
BrowseName	ActuatorClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTConditionClassType (See Data Items Documentation)					

9.10.2 Defintion of `CommunicationsClassType`

Table 229: `CommunicationsClassType` Definition

Attribute	Value				
BrowseName	CommunicationsClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTConditionClassType (See Data Items Documentation)					

9.10.3 Defintion of `DataRangeClassType`

Table 230: `DataRangeClassType` Definition

Attribute	Value				
BrowseName	DataRangeClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTConditionClassType (See Data Items Documentation)					

2582 9.10.4 Defintion of `HardwareClassType`

Table 231: `HardwareClassType` Definition

Attribute	Value				
BrowseName	HardwareClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTConditionClassType (See Data Items Documentation)					

2583 9.10.5 Defintion of `LogicProgramClassType`

Table 232: `LogicProgramClassType` Definition

Attribute	Value				
BrowseName	LogicProgramClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTConditionClassType (See Data Items Documentation)					

2584 9.10.6 Defintion of `MotionProgramClassType`

Table 233: `MotionProgramClassType` Definition

Attribute	Value				
BrowseName	MotionProgramClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTConditionClassType (See Data Items Documentation)					

2585 9.10.7 Defintion of `SystemClassType`

Table 234: `SystemClassType` Definition

Attribute	Value				
BrowseName	SystemClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTConditionClassType (See Data Items Documentation)					

2586 9.11 Data Item Sub Types

2587 The `subTypes` are related to the various ***Variables*** using the **`HasMTSubClassType`**.
2588 These provide the the second level type classifiaction of the MTConnect *DataItems* and
2589 also the **`ConditionSubClassId`** for the `Conditions`.

2590 9.11.1 Defintion of `MTDataItemSubClassType`

Table 235: `MTDataItemSubClassType` Definition

Attribute	Value				
BrowseName	MTDataItemSubClassType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of BaseConditionClassType (See [UA Part 09] Documentation)					
HasSubtype	ObjectType	RelativeSubClassType		See section 9.11.50	
HasSubtype	ObjectType	RemainingSubClassType		See section 9.11.51	
HasSubtype	ObjectType	RequestSubClassType		See section 9.11.52	
HasSubtype	ObjectType	ResponseSubClassType		See section 9.11.53	
HasSubtype	ObjectType	RockwellSubClassType		See section 9.11.54	
HasSubtype	ObjectType	RotarySubClassType		See section 9.11.55	
HasSubtype	ObjectType	SetUpSubClassType		See section 9.11.56	
HasSubtype	ObjectType	ShoreSubClassType		See section 9.11.57	
HasSubtype	ObjectType	StandardSubClassType		See section 9.11.58	
HasSubtype	ObjectType	SwitchedSubClassType		See section 9.11.59	
HasSubtype	ObjectType	TargetSubClassType		See section 9.11.60	
HasSubtype	ObjectType	ToolChangeStopSubClassType		See section 9.11.61	
HasSubtype	ObjectType	ToolEdgeSubClassType		See section 9.11.62	
HasSubtype	ObjectType	ToolGroupSubClassType		See section 9.11.63	
HasSubtype	ObjectType	ToolSubClassType		See section 9.11.64	
HasSubtype	ObjectType	UasbleSubClassType		See section 9.11.65	
HasSubtype	ObjectType	VerticalSubClassType		See section 9.11.66	
HasSubtype	ObjectType	VolumeSubClassType		See section 9.11.67	
HasSubtype	ObjectType	VickersSubClassType		See section 9.11.68	
HasSubtype	ObjectType	WeightSubClassType		See section 9.11.69	
HasSubtype	ObjectType	WorkingSubClassType		See section 9.11.70	
HasSubtype	ObjectType	WorkpieceSubClassType		See section 9.11.71	
Continued...					

References	NodeClass	BrowseName	DataType	TypeDefinition	ModelingRule
HasSubtype	ObjectType	LineSubClassType		See section 9.11.28	
HasSubtype	ObjectType	LoadedSubClassType		See section 9.11.29	
HasSubtype	ObjectType	MachineAxisLockSubClassType		See section 9.11.30	
HasSubtype	ObjectType	MaintenanceSubClassType		See section 9.11.31	
HasSubtype	ObjectType	ManualUnclampSubClassType		See section 9.11.32	
HasSubtype	ObjectType	MaximumSubClassType		See section 9.11.33	
HasSubtype	ObjectType	MinimumSubClassType		See section 9.11.34	
HasSubtype	ObjectType	MohsSubClassType		See section 9.11.35	
HasSubtype	ObjectType	MoleSubClassType		See section 9.11.36	
HasSubtype	ObjectType	MotionSubClassType		See section 9.11.37	
HasSubtype	ObjectType	NoScaleSubClassType		See section 9.11.38	
HasSubtype	ObjectType	OperatingSubClassType		See section 9.11.39	
HasSubtype	ObjectType	OperatorSubClassType		See section 9.11.40	
HasSubtype	ObjectType	OptionalStopSubClassType		See section 9.11.41	
HasSubtype	ObjectType	OverrideSubClassType		See section 9.11.42	
HasSubtype	ObjectType	PrimarySubClassType		See section 9.11.43	
HasSubtype	ObjectType	PoweredSubClassType		See section 9.11.44	
HasSubtype	ObjectType	ProbeSubClassType		See section 9.11.45	
HasSubtype	ObjectType	ProcessSubClassType		See section 9.11.46	
HasSubtype	ObjectType	ProgrammedSubClassType		See section 9.11.47	
HasSubtype	ObjectType	RadialSubClassType		See section 9.11.48	
HasSubtype	ObjectType	RapidSubClassType		See section 9.11.49	
Continued...					

References	NodeClass	BrowseName	DataType	TypeDefinition	ModelingRule
HasSubtype	ObjectType	AlternatingSubClassType		See section 9.11.6	
HasSubtype	ObjectType	AScaleSubClassType		See section 9.11.7	
HasSubtype	ObjectType	AuxiliarySubClassType		See section 9.11.8	
HasSubtype	ObjectType	BadSubClassType		See section 9.11.9	
HasSubtype	ObjectType	BrinellSubClassType		See section 9.11.10	
HasSubtype	ObjectType	BScaleSubClassType		See section 9.11.11	
HasSubtype	ObjectType	CommandedSubClassType		See section 9.11.12	
HasSubtype	ObjectType	GoodSubClassType		See section 9.11.13	
HasSubtype	ObjectType	ControlSubClassType		See section 9.11.14	
HasSubtype	ObjectType	CScaleSubClassType		See section 9.11.15	
HasSubtype	ObjectType	DelaySubClassType		See section 9.11.16	
HasSubtype	ObjectType	DirectSubClassType		See section 9.11.17	
HasSubtype	ObjectType	DryRunSubClassType		See section 9.11.18	
HasSubtype	ObjectType	DScaleSubClassType		See section 9.11.19	
HasSubtype	ObjectType	FixtureSubClassType		See section 9.11.20	
HasSubtype	ObjectType	IncrementalSubClassType		See section 9.11.21	
HasSubtype	ObjectType	JobSubClassType		See section 9.11.22	
HasSubtype	ObjectType	KineticSubClassType		See section 9.11.23	
HasSubtype	ObjectType	LateralSubClassType		See section 9.11.24	
HasSubtype	ObjectType	LeebSubClassType		See section 9.11.25	
HasSubtype	ObjectType	LengthSubClassType		See section 9.11.26	
HasSubtype	ObjectType	LinearSubClassType		See section 9.11.27	
Continued...					

References	NodeClass	BrowseName	DataType	TypeDefinition	ModelingRule
HasSubtype	ObjectType	AbsoluteSubClassType		See section 9.11.2	
HasSubtype	ObjectType	ActualSubClassType		See section 9.11.3	
HasSubtype	ObjectType	ActionSubClassType		See section 9.11.4	
HasSubtype	ObjectType	AllSubClassType		See section 9.11.5	

2591 9.11.2 Defintion of AbsoluteSubClassType

2592 The magnitude or measurement of a type irrespective of its relation to other values.

Table 236: AbsoluteSubClassType Definition

Attribute	Value				
BrowseName	AbsoluteSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2593 9.11.3 Defintion of ActualSubClassType

2594 The measured value of the a type.

Table 237: ActualSubClassType Definition

Attribute	Value				
BrowseName	ActualSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2595 9.11.4 Defintion of ActionSubClassType

2596 An indication of the operating state or value of a type.

Table 238: ActionSubClassType Definition

Attribute	Value				
BrowseName	ActionSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2597 9.11.5 Defintion of AllSubClassType

Table 239: `AllSubClassType` Definition

Attribute	Value				
BrowseName	AllSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.6 Defintion of `AlternatingSubClassType`

The measurement of a type occurring in turn repeatedly.

Table 240: `AlternatingSubClassType` Definition

Attribute	Value				
BrowseName	AlternatingSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.7 Defintion of `AScaleSubClassType`

A Scale weighting factor for the measurement of sound level.

Table 241: `AScaleSubClassType` Definition

Attribute	Value				
BrowseName	AScaleSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.8 Defintion of `AuxiliarySubClassType`

Example: When multiple locations on a piece of bar stock are referenced as the indication for the `END_OF_BAR`, the additional location(s) MUST be designated as `AUXILIARY` indication(s) for the `END_OF_BAR`.

Table 242: `AuxiliarySubClassType` Definition

Attribute	Value				
BrowseName	AuxiliarySubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2606 9.11.9 Defintion of `BadSubClassType`

2607 Indicates the count of incorrect parts produced.

Table 243: `BadSubClassType` Definition

Attribute	Value				
BrowseName	BadSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2608 9.11.10 Defintion of `BrinellSubClassType`

2609 A scale to measure the resistance to deformation of a surface.

Table 244: `BrinellSubClassType` Definition

Attribute	Value				
BrowseName	BrinellSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2610 9.11.11 Defintion of `BScaleSubClassType`

2611 B Scale weighting factor for the measurement of sound level.

Table 245: `BScaleSubClassType` Definition

Attribute	Value				
BrowseName	BScaleSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.12 Defintion of `CommandedSubClassType`

The value as specified by the Controller type component.

Table 246: `CommandedSubClassType` Definition

Attribute	Value				
BrowseName	CommandedSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.13 Defintion of `GoodSubClassType`

Indicates the count of correct parts made.

Table 247: `GoodSubClassType` Definition

Attribute	Value				
BrowseName	GoodSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.14 Defintion of `ControlSubClassType`

The state of the enabling signal or control logic that enables or disables the function or operation of the *Structural Element*.

Table 248: `ControlSubClassType` Definition

Attribute	Value				
BrowseName	ControlSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2619 9.11.15 Defintion of `CScaleSubClassType`

2620 C Scale weighting factor for the measurement of sound level.

Table 249: `CScaleSubClassType` Definition

Attribute	Value				
BrowseName	CScaleSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2621 9.11.16 Defintion of `DelaySubClassType`

2622 Measurement of the time that a piece of equipment is waiting for an event or an action to
2623 occur.

Table 250: `DelaySubClassType` Definition

Attribute	Value				
BrowseName	DelaySubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2624 9.11.17 Defintion of `DirectSubClassType`

2625 Measurement of DC current or voltage.

Table 251: `DirectSubClassType` Definition

Attribute	Value				
BrowseName	DirectSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.18 Defintion of `DryRunSubClassType`

A setting or operator selection used to execute a test mode to confirm the execution of machine functions.

Table 252: `DryRunSubClassType` Definition

Attribute	Value				
BrowseName	DryRunSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.19 Defintion of `DScaleSubClassType`

D Scale weighting factor for the measurement of sound level.

Table 253: `DScaleSubClassType` Definition

Attribute	Value				
BrowseName	DScaleSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.20 Defintion of `FixtureSubClassType`

Fixture denotes a specifc type of a piece of equipment.

Table 254: `FixtureSubClassType` Definition

Attribute	Value				
BrowseName	FixtureSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.21 Defintion of `IncrementalSubClassType`

A small change which could be either positive or negative in a Type's value or function.

Example: The position of a block of program code relative to the occurrence of the last `LINE_LABEL` encountered in the control program.

Table 255: `IncrementalSubClassType` Definition

Attribute	Value				
BrowseName	IncrementalSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.22 Defintion of `JobSubClassType`

The value of a signal or calculation issued to adjust the feedrate of the axes associated with a Path component when the axes, or a single axis, are being operated in a manual mode or method (jogging).

Table 256: `JobSubClassType` Definition

Attribute	Value				
BrowseName	JobSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.23 Defintion of `KineticSubClassType`

Table 257: `KineticSubClassType` Definition

Attribute	Value				
BrowseName	KineticSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2642 9.11.24 Defintion of `LateralSubClassType`

2643 An indication of the position of a mechanism that may move in a lateral direction. The
2644 mechanism is represented by a `Composition` type component.

Table 258: `LateralSubClassType` Definition

Attribute	Value				
BrowseName	LateralSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2645 9.11.25 Defintion of `LeebSubClassType`

2646 A scale to measure the elasticity of a surface.

Table 259: `LeebSubClassType` Definition

Attribute	Value				
BrowseName	LeebSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2647 9.11.26 Defintion of `LengthSubClassType`

2648 The measurement or extent of something from end to end.

Table 260: `LengthSubClassType` Definition

Attribute	Value				
BrowseName	LengthSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2649 9.11.27 Defintion of `LinearSubClassType`

2650 The direction of motion.

Table 261: `LinearSubClassType` Definition

Attribute	Value				
BrowseName	LinearSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2651 9.11.28 Defintion of `LineSubClassType`

2652 The state of the power source for the *Structural Element*.

Table 262: `LineSubClassType` Definition

Attribute	Value				
BrowseName	LineSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2653 9.11.29 Defintion of `LoadedSubClassType`

2654 An indication that the sub-parts of a piece of equipment are under load.

Table 263: `LoadedSubClassType` Definition

Attribute	Value				
BrowseName	LoadedSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.30 Defintion of `MachineAxisLockSubClassType`

A setting or operator selection that changes the behavior of the controller on a piece of equipment.

Table 264: `MachineAxisLockSubClassType` Definition

Attribute	Value				
BrowseName	MachineAxisLockSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.31 Defintion of `MaintenanceSubClassType`

The identifier of the person currently responsible for performing maintenance on the piece of equipment.

Table 265: `MaintenanceSubClassType` Definition

Attribute	Value				
BrowseName	MaintenanceSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.32 Defintion of `ManualUnclampSubClassType`

An indication of the state of an operator controlled interlock that can inhibit the ability to initiate an unclamp action of an electronically controlled chuck.

Table 266: `ManualUnclampSubClassType` Definition

Attribute	Value				
BrowseName	ManualUnclampSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2664 9.11.33 Defintion of MaximumSubClassType

2665 Maximum or peak value recorded for the data item during the calculation period.

Table 267: `MaximumSubClassType` Definition

Attribute	Value				
BrowseName	MaximumSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2666 9.11.34 Defintion of MinimumSubClassType

2667 Minimum value recorded for the data item during the calculation period.

Table 268: `MinimumSubClassType` Definition

Attribute	Value				
BrowseName	MinimumSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2668 9.11.35 Defintion of MohsSubClassType

2669 A scale to measure the resistance to scratching of a surface.

Table 269: `MohsSubClassType` Definition

Attribute	Value				
BrowseName	MohsSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.36 Defintion of `MoleSubClassType`

Table 270: `MoleSubClassType` Definition

Attribute	Value				
BrowseName	MoleSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.37 Defintion of `MotionSubClassType`

An indication of the open or closed state of a mechanism. The mechanism is represented by a `Composition` type component.

Table 271: `MotionSubClassType` Definition

Attribute	Value				
BrowseName	MotionSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.38 Defintion of `NoScaleSubClassType`

No weighting factor on the frequency scale for the measurement of sound level.

Table 272: `NoScaleSubClassType` Definition

Attribute	Value				
BrowseName	NoScaleSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.39 Defintion of `OperatingSubClassType`

An indication that the major sub-parts of a piece of equipment are powered or performing any activity whether producing a part or product or not.

Example: For traditional machine tools, this includes when the piece of equipment is `WORKING` or it is idle.

Table 273: `OperatingSubClassType` Definition

Attribute	Value				
BrowseName	OperatingSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.40 Defintion of `OperatorSubClassType`

The identifier of the person currently responsible for operating the piece of equipment.

Table 274: `OperatorSubClassType` Definition

Attribute	Value				
BrowseName	OperatorSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.41 Defintion of `OptionalStopSubClassType`

A setting or operator selection that changes the behavior of the controller on a piece of equipment.

Table 275: `OptionalStopSubClassType` Definition

Attribute	Value				
BrowseName	OptionalStopSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.42 Defintion of `OverrideSubClassType`

The operator's overridden value.

Table 276: `OverrideSubClassType` Definition

Attribute	Value				
BrowseName	OverrideSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.43 Defintion of `PrimarySubClassType`

Specific applications MAY reference one or more locations on a piece of bar stock as the indication for the `END_OF_BAR`. The main or most important location MUST be designated as the `PRIMARY` indication for the `END_OF_BAR`.

Table 277: `PrimarySubClassType` Definition

Attribute	Value				
BrowseName	PrimarySubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.44 Defintion of `PoweredSubClassType`

An indication that primary power is applied to the piece of equipment and, as a minimum, the controller or logic portion of the piece of equipment is powered and functioning or components that are required to remain on are powered.

Table 278: `PoweredSubClassType` Definition

Attribute	Value				
BrowseName	PoweredSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.45 Defintion of `ProbeSubClassType`

The position provided by a measurement probe.

Table 279: `ProbeSubClassType` Definition

Attribute	Value				
BrowseName	ProbeSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.46 Defintion of `ProcessSubClassType`

The measurement of the time from the beginning of production of a part or product on a piece of equipment until the time that production is complete for that part or product on that piece of equipment. This includes the time that the piece of equipment is running, producing parts or products, or in the process of producing parts.

Table 280: `ProcessSubClassType` Definition

Attribute	Value				
BrowseName	ProcessSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.47 Defintion of `ProgrammedSubClassType`

The value of a signal or calculation issued to adjust the feedrate of the axes associated with a `Path` component when the axes, or a single axis, are operating as specified by a logic or motion program or set by a switch.

Table 281: `ProgrammedSubClassType` Definition

Attribute	Value				
BrowseName	ProgrammedSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.48 Defintion of `RadialSubClassType`

A reference to a radial type tool offset variable.

Table 282: `RadialSubClassType` Definition

Attribute	Value				
BrowseName	RadialSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.49 Defintion of `RapidSubClassType`

The value of a signal or calculation issued to adjust the feedrate of the axes associated with a `Path` component when the axes, or a single axis, are being operated in a rapid positioning mode or method (rapid).

Table 283: `RapidSubClassType` Definition

Attribute	Value				
BrowseName	RapidSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.50 Defintion of `RelativeSubClassType`

Table 284: `RelativeSubClassType` Definition

Attribute	Value				
BrowseName	RelativeSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.51 Defintion of `RemainingSubClassType`

The remaining amount of the type specified.

Table 285: `RemainingSubClassType` Definition

Attribute	Value				
BrowseName	RemainingSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.52 Defintion of `RequestSubClassType`

`Request` subtype identifies if the data item defined for MTConnect Interaction Model [MTConnect Part 5.0] represents a request.

Table 286: `RequestSubClassType` Definition

Attribute	Value				
BrowseName	RequestSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.53 Defintion of `ResponseSubClassType`

`Response` subtype identifies if the data item defined for MTConnect Interaction Model [MTConnect Part 5.0] represents a response.

Table 287: `ResponseSubClassType` Definition

Attribute	Value				
BrowseName	ResponseSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2722 9.11.54 Defintion of `RockwellSubClassType`

2723 A scale to measure the resistance to deformation of a surface.

Table 288: `RockwellSubClassType` Definition

Attribute	Value				
BrowseName	RockwellSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2724 9.11.55 Defintion of `RotarySubClassType`

2725 The rotational direction of a rotary motion using the right hand rule convention.

Table 289: `RotarySubClassType` Definition

Attribute	Value				
BrowseName	RotarySubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2726 9.11.56 Defintion of `SetUpSubClassType`

2727 The identifier of the person currently responsible for operating the piece of equipment.

Table 290: `SetUpSubClassType` Definition

Attribute	Value				
BrowseName	SetUpSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2728 9.11.57 Defintion of `ShoreSubClassType`

2729 A scale to measure the resistance to deformation of a surface.

Table 291: `ShoreSubClassType` Definition

Attribute	Value				
BrowseName	ShoreSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2730 9.11.58 Defintion of `StandardSubClassType`

2731 The standard or original value of an object.

Table 292: `StandardSubClassType` Definition

Attribute	Value				
BrowseName	StandardSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2732 9.11.59 Defintion of `SwitchedSubClassType`

2733 An indication of the activation state of a mechanism represented by a `Composition`
2734 type component.

2735 The activation state indicates whether the `Composition` element is activated or not.

Table 293: `SwitchedSubClassType` Definition

Attribute	Value				
BrowseName	SwitchedSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.60 Defintion of `TargetSubClassType`

Indicates the number of parts that are projected or planned to be produced.

Table 294: `TargetSubClassType` Definition

Attribute	Value				
BrowseName	TargetSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.61 Defintion of `ToolChangeStopSubClassType`

A setting or operator selection that changes the behavior of the controller on a piece of equipment.

Table 295: `ToolChangeStopSubClassType` Definition

Attribute	Value				
BrowseName	ToolChangeStopSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.62 Defintion of `ToolEdgeSubClassType`

Table 296: `ToolEdgeSubClassType` Definition

Attribute	Value				
BrowseName	ToolEdgeSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2742 9.11.63 Defintion of `ToolGroupSubClassType`

2743 The tool group a specific tool is assigned to in the part program.

Table 297: `ToolGroupSubClassType` Definition

Attribute	Value				
BrowseName	ToolGroupSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2744 9.11.64 Defintion of `ToolSubClassType`

Table 298: `ToolSubClassType` Definition

Attribute	Value				
BrowseName	ToolSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2745 9.11.65 Defintion of `UasbleSubClassType`

2746 The remaining useable value of an object.

Table 299: `UasbleSubClassType` Definition

Attribute	Value				
BrowseName	UasbleSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.66 Defintion of `VerticalSubClassType`

An indication of the position of a mechanism that may move in a vertical direction.

Table 300: `VerticalSubClassType` Definition

Attribute	Value				
BrowseName	VerticalSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.67 Defintion of `VolumeSubClassType`

A measurement of space accupied by a physical object.

Table 301: `VolumeSubClassType` Definition

Attribute	Value				
BrowseName	VolumeSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

9.11.68 Defintion of `VickersSubClassType`

A scale to measure the resistance to deformation of a surface.

Table 302: `VickersSubClassType` Definition

Attribute	Value				
BrowseName	VickersSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2753 9.11.69 Defintion of `WeightSubClassType`

2754 A physical object's relative mass.

Table 303: `WeightSubClassType` Definition

Attribute	Value				
BrowseName	WeightSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2755 9.11.70 Defintion of `WorkingSubClassType`

2756 An indication that a piece of equipment is performing any activity.

Table 304: `WorkingSubClassType` Definition

Attribute	Value				
BrowseName	WorkingSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2757 9.11.71 Defintion of `WorkpieceSubClassType`

2758 A physical object being or to be worked on with a tool or machine.

Table 305: `WorkpieceSubClassType` Definition

Attribute	Value				
BrowseName	WorkpieceSubClassType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of MTDataItemSubClassType (See section 9.11.1)					

2759 9.12 MTConnect Device Profile

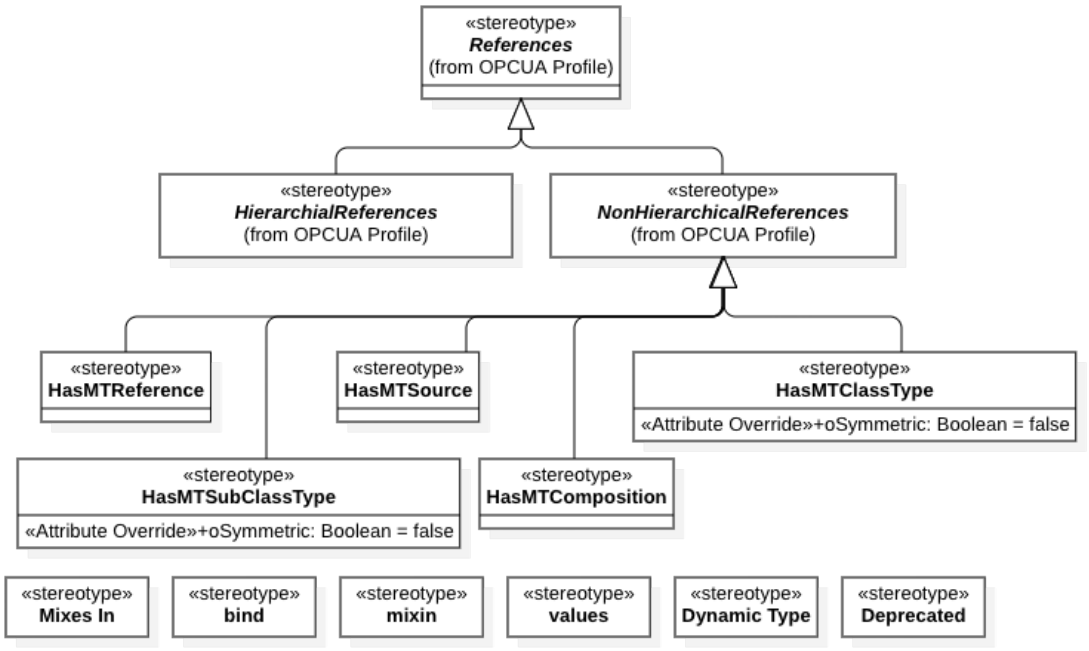

Figure 36: MTConnect Device Profile Diagram

2760 The device profile documents the common data types and stereotypes that are used to
2761 construct the model. A stereotype is a design or modeling pattern that provides additional
2762 information about the type or the relationship between types.

2763 It can also identify the behavior of a property or the role the type or relation will play in
2764 the model.

2765 Stereotypes are used throughout the model to provide additional information that will halp
2766 provide context and definition to aid in better understanding the data model.

2767 9.12.1 Defintion of «Deprecated»

2768 An MTConnect deprecated entity that is kept for backward compatibility.

9.12.2 Defintion of «Dynamic Type»

A dynamic type indicates that the class browse name and type will be determined at configuration time based on the MTConnect *Device* elements.

9.12.3 Defintion of «HasMTClassType»

A *DataItem* is representated in OPC UA as a sub-type of the most appropriate **BaseDataVariableType**. The type is derived from the MTConnect `type` attribute and references the corect `..ClassType`.

Table 306: «HasMTClassType» Definition

Attribute	Value				
BrowseName	HasMTClassType				
IsAbstract	False				
Symmetric	false				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of NonHierarchicalReferences (See [UA Part 05] Documentation)					

9.12.4 Defintion of «HasMTComposition»

A reference from the MTConnect Data Item or condition to the Composition Object of the Component.

Table 307: «HasMTComposition» Definition

Attribute	Value				
BrowseName	HasMTComposition				
IsAbstract	False				
Symmetric	true				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of NonHierarchicalReferences (See [UA Part 05] Documentation)					

9.12.5 Defintion of «HasMTReference»

A reference from one component to either a *DataItem* or a *Component*.

Table 308: «HasMTReference» Definition

Attribute	Value				
BrowseName	HasMTReference				
IsAbstract	False				
Symmetric	true				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of NonHierarchicalReferences (See [UA Part 05] Documentation)					

9.12.6 Defintion of «HasMTSource»

The Source relation to a *Component* or *DataItem*.

Table 309: «HasMTSource» Definition

Attribute	Value				
BrowseName	HasMTSource				
IsAbstract	False				
Symmetric	true				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of NonHierarchicalReferences (See [UA Part 05] Documentation)					

9.12.7 Defintion of «HasMTSubClassType»

A *DataItem* is representated in OPC UA as a sub-type of the most appropriate **BaseDataVariableType**. The sub-type is derived from the MTConnect subType attribute and references the corect ..ClassType.

Table 310: «HasMTSubClassType» Definition

Attribute	Value				
BrowseName	HasMTSubClassType				
IsAbstract	False				
Symmetric	false				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of NonHierarchicalReferences (See [UA Part 05] Documentation)					

9.12.8 Defintion of «Mixes In»

This stereotype is associated with the dependency between a type and a mixin. See Section 9.12.10 for a complete description of the mixin.

9.12.9 Defintion of «bind»

When a dynamic type (See Section 9.12.2) creates an instance where the super-type can be associated based on the data item category and type, the `Type Factory` will specify which supertype is to be referenced.

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

9.12.10 Defintion of «mixin»

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

9.12.11 Defintion of «use»

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

9.12.12 Defintion of «values»

For controlled vocabularies of enumerated types, specifies the relationship to the allowable values for the type.

10 Profiles and Namespaces

10.1 Namespace Metadata

Table 311 defines the ***namespace metadata*** for this specification. The ***Object*** is used to provide information for the ***namespace*** and an indication about static ***Nodes***. Static ***Nodes*** are identical for all ***Attributes*** in all ***Servers***, including the ***Value Attribute***. See [UA Part 05] for more details.

The information is provided as ***Object*** of type **`NamespaceMetadataType`**. This ***Object*** is a component of the ***Namespaces Object*** that is part of the ***Server Object***. The **`NamespaceMetadataType ObjectType`** and its ***Properties*** are defined in [UA Part 05].

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

Table 311: `NamespaceMetadata` *Object* for this Specification

Attribute	**Value**		
BrowseName	http://opcfoundation.org/UA/MTConnect/v2/		
References	**BrowseName**	**DataType**	**Value**
HasProperty	NamespaceUri	String	http://opcfoundation.org/UA/MTConnect/v2/
HasProperty	NamespaceVersion	String	2.0
HasProperty	NamespacePublicationDate	DateTime	2018-10-31T00:00:00
HasProperty	IsNamespaceSubset	Boolean	False
HasProperty	StaticNodeIdTypes	IdType[]	[0]
HasProperty	StaticNumericNodeIdRange	NumericRange[]	[1:1073741824]
HasProperty	StaticStringNodeIdPattern	String	

10.2 Conformance Units and Profiles

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

10.2.1 Server

Table 312 defines the ***Server*** based ***ConformanceUnits***.

Table 312: MTConnect ***Server*** Information Model

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

10.2.2 Client

Table 313 defines the ***Client*** based ***ConformanceUnits***.

Table 313: MTConnect ***Client*** Information Model

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

10.3 Handling of OPC UA Namespaces

Namespaces are used by OPC UA to create unique identifiers across different naming authorities. The Attributes **`NodeId`** and **`BrowseName`** are identifiers. A ***Node*** in the UA ***AddressSpace*** is unambiguously identified using a NodeId. Unlike **`NodeIds`**, the **`BrowseName`** cannot be used to unambiguously identify a ***Node***. Different ***Nodes*** may

have the same **BrowseName**. They are used to build a browse path between two Nodes or to define a standard ***Property***.

Servers may often choose to use the same namespace for the **NodeId** and the **BrowseName**. However, if they want to provide a standard ***Property***, its glsBrowseName shall have the ***namespace*** of the standards body although the ***namespace*** of the **NodeId** reflects something else, for example the **EngineeringUnits** ***Property***. All **NodeIds** of ***Nodes*** not defined in this specification shall not use the standard ***namespaces***.

Table 314 provides a list of mandatory and optional namespaces used in an MTConnect OPC UA ***Server***.

Table 314: Namespaces used in a MTConnect Server

NamespaceURI	Description	Use
http://opcfoundation.org/UA/	***Namespace*** for **NodeIds** and **BrowseNames** defined in the OPC UA specification. This ***namespace*** shall have ***namespace*** index 0.	Mandatory
Local Server URI	***Namespace*** for nodes defined in the local server. This may include types and instances used in an ***AutoID*** Device represented by the ***Server***. This ***namespace*** shall have ***namespace*** index 1.	Mandatory
http://opcfoundation.org/UA/MTConnect/v2/	***Namespace*** for **NodeIds** and **BrowseNames** defined in this specification. The ***namespace*** index is ***Server*** specific.	Mandatory
Vendor specific types	A ***Server*** may provide vendor-specific types like types derived from ***ObjectTypes*** defined in this specification in a vendor-specific ***namespace***.	Optional
Vendor specific instances	A ***Server*** provides vendor-specific instances of the standard types or vendor-specific instances of vendor-specific types in a vendor-specific ***namespace***. It is recommended to separate vendor specific types and vendor specific instances into two or more ***namespaces***.	Mandatory

Table 315 provides a list of ***namespaces*** and their index used for **BrowseNames** in this specification. The default ***namespace*** of this specification is not listed since all **BrowseNames** without prefix use this default ***namespace***.

Table 315: Namespaces used used in this specification

NamespaceURI	Namespace Index	Example
http://opcfoundation.org/UA/	0	0:EngineeringUnits
http://opcfoundation.org/UA/MTConnect/v2/	1	1:MTDevice

Annex A MTConnect Namespace and Mappings (normative)

A.1 Namespace and identifiers for MTConnect Information Model

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

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

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

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

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

http://www.opcfoundation.org/UA/schemas/MTConnect/v2/2.00/MTConnect.NodeIds.csv

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

http://www.opcfoundation.org/UA/schemas/MTConnect/v2/MTConnect.NodeIds.csv

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

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

http://opcfoundation.org/UA/schemas/MTConnect/v2/2.00/Opc.Ua.MTConnect.NodeSet2.xml

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

http://opcfoundation.org/UA/schemas/MTConnect/v2/Opc.Ua.MTConnect.NodeSet2.xml