F O U N D A T I O N

MTConnect®

OPC 30070-1

OPC UA for MTConnect®

Amendment 1: Conditions

Release 2.00.01

June 5, 2020

OPC UA Companion Specification – Amendment

June 5, 2020

Specification Type:	Industry Standard Specification	Comments:	
Document Number	**OPC 30070-1**		
Title:	OPC UA for MTConnect® Amendment 1: Conditions	Date:	June 5, 2020
Version:	Release 2.00.01	Software:	LaTeX
Authors:	William Sobel	Source:	OPC 30070 – UA CS for MTConnect 2.00 – Amendment 1 - Conditions.pdf
Owner:	MTConnect Institute	Status:	Release

Document History

Version	**Date**	**Reason**	**Comments**	**Mantis**
2.00.01	2019-11-01	Revision	Mapping the Native Code to a Condition Branch Semantic Correction	4883

Contents

List of Figures

List of Tables

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 Foundation 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 Foundation 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 Foundation 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.

OPC 30070-1 – OPC UA for MTConnect®

Amendment 1: Conditions

OPC UA for MTConnect - Part 1: Device Model
Clause 8.4.6: Replace 8.4.6 with the following:

Listing 7: Controller and Path Components and Their Data Items

```
<Controller id="p5add360">
  <DataItems>
    <DataItem id="x7ca94e0" type="EMERGENCY_STOP"
       category="EVENT" name="estop"/>
    <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>
```

8.4.6 Conditions

In [MTConnect Part 2.0], the `DataItem` represents the metadata describing the semantic meaning of the `Condition` as it relates to its component using an object instance of type `MTConditionType`. The activation and state of `Conditions` is represented by the `MTConditionEventType` that is a subtype of the **`BaseConditionType`**. The MTConnect `Conditions` in [MTConnect Part 3.0] is 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* respectively. 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.

When a `Condition` becomes active in MTConnect, it will transition from `Normal` to `Warning` or `Fault` state. The transition will cause an **`Event`** to be dispatched of the `MTConditionEventType`. The `MTConditionEventType` has a ***Property*** called **`ActiveState`** that indicates that it is 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`**. An active `Condition` will require the **`Retain`** flag of the `MTConditionEventType` instance to be **`True`**.

Listing 10: Rotary C Component Stream

```
<ComponentStream componentId="zf476090" component="Rotary" name="
   C" nativeName="S">
  <Condition>
    <Normal sequence="201" timestamp="2018-10-31T20:34:19.9981Z"
   dataItemId="afb596b0" type="AMPERAGE" compositionId="b7792870
   " name="Soverload"/>
    <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>
    <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>
  </Condition>
  ...
```

Each time an MTConnect `Condition` activates or deactivates, a ***Condition*** **`Event`** will be reported associated with the meta-data instance of the `MTConditionType` using the **`NodeId`** as the **`SourceNode`** of the **`Event`**. `MTConditionEventType` is a subtype of the **`Event`** and MUST never be instantiated in the address space as an ***Object***.

8.4.6.1 Mapping Conditions

MTConnect allows Conditions to represent multiple instances simultaneous *Faults* and *Warnings* associated with a Component and of a particular *Type*. In MTConnect a *Type* can be something like a TEMPERATURE or a LOGICAL_PROGRAM. The Conditions *Faults* and *Warnings* are associated by their unique characteristics of their description or more commonly their nativeCode.

Every time a condition is reported as a separate instance, as described in [MTConnect Part 3.0], it is considered another activation of the *Condition* and will be associated with a unique **ConditionId** as the specific **NodeId** of the MTConditionEventType. The **ConditionName** is handled in the same manner as the **ConditionId** and must be unique for a stream of associated *Condition* set of states. 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 for a current request. When all active *Condition*s are reported as Normal, an MTConditionEventType for each active *Condition* must be reported with the ActiveState set to **False** and the **Retain** set to **False**.

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

Table 12: Mapping to `MTConditionEventType` Properties

Property	**Type**	**Mapping**
(Attribute) NodeId	NodeId	A **`NodeId`** associated with the MTConnect *Condition* stream. Often given by the `nativeCode` attribute. Referred to as the **`ConditionId`**
EventId	ByteString	Auto-generated by the server per [UA Part 05]
EventType	NodeId	The **`NodeId`** of the `MTConditionEventType`.
SourceNode	NodeId	The **`NodeId`** of the ***Instance*** of the `MTConditionType` ***Object*** representing the `DataItem` with `category` `CONDITION`.
SourceName	NodeId	The **`BrowseName`** of the **`SourceNode`** referenced above.
Time	UtcTime	From MTConnect timestamp attribute.
ReceiveTime	UtcTime	Current time when MTConnect Condition received by OPC UA Server.
LocalTime	TimeZoneDataType	Optionally supplied by OPC UA Server since MTConnect uses UTC.
Message	LocalizedText	MTConnect Condition `CDATA`.
Severity	UInt16	Taking the value for the *QName* of the `Condition`: • When `Normal`, **`Severity`** is 0. • When `Warning`, **`Severity`** is 500. • When `Fault`, **`Severity`** is 1000.
ConditionClassId	NodeId	The **`NodeId`** for the `ClassType` representing the `type` attribute of the `DataItem`.
ConditionClassName	LocalizedText	The name associated with the **`ConditionClassId`**.
ConditionSubClassId	NodeId	The **`NodeId`** for the `ClassType` representing the `subType` attribute of the `DataItem`.
ConditionSubClassName	LocalizedText	The name associated with the **`ConditionSubClassId`**.
ConditionName	String	A text version of the set of associated conditions, should follow the same rules as the **`ConditionId`**. For example, the name MAY be composed of the `SourceName` and the `nativeCode`.
BranchId	NodeId	Not used for MTConnect.
Retain	Boolean	Taking the value for the *QName* of the `Condition`: • When only `Normal`, **`False`** • When `Warning` or `Fault`, **`True`**
EnabledState	LocalizedText	Taking the value for the *QName* of the `Condition`: • When `Unavailable`, **`Disabled`** • Otherwise, **`Enabled`**
Quality	StausCode	Taking the value for the *QName* of the `Condition`: • When `Unavailable`, **`Bad_NotConnected`** • Otherwise, **`Good`**
LastSeverity	Uint16	Set to the previous severity for this condition.
Comment	LocalizedText	Set to the *CDATA* of the Condition.
ClientUserId	String	The `name` of the `Device`.
ActiveState	LocalizedText	Taking the value for the *QName* of the `Condition`: • When only `Normal`, **`Inactive`** • When `Warning` or `Fault`, **`Active`**

8.4.6.2 MTConnect Condition Parallel Activation

As stated above, MTConnect allows for multiple Conditions of the same type to be active at the same time. In MTConnect the conditions are differentiated by their nativeCode or CDATA. In the following example, the attribute nativeCode is used to indicate the independent activations of the Condition. In this diagram, there are three activations–0, 1, and 2–of the PLC alarms and are associated with the LOGIC_PROGRAM. Each of the unique nativeCodes is mapped to an **Event** and tracked separately.

The Condition is instantiated in the **AddressSpace** as an MTConditionType and acts as the source of the **Event**s when they are sent. The individual activation and deactivations are tracked using the **Event** mechanism as described in [UA Part 03] and [UA Part 04]. All **Event**s will have the same **SourceNode** since they are all produced from the same MTConditionType instance.

Table 13 represents the state transitions of the key OPC UA **Condition** model and the reporting of MTConditionEventType. The text that follows will refer to this table and the MTConnect Extensible Markup Language (XML) to illustrate the expected behavior. Figure 30 gives a visual representation of the event reporting.

Table 13: LogicProgramCondition States

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

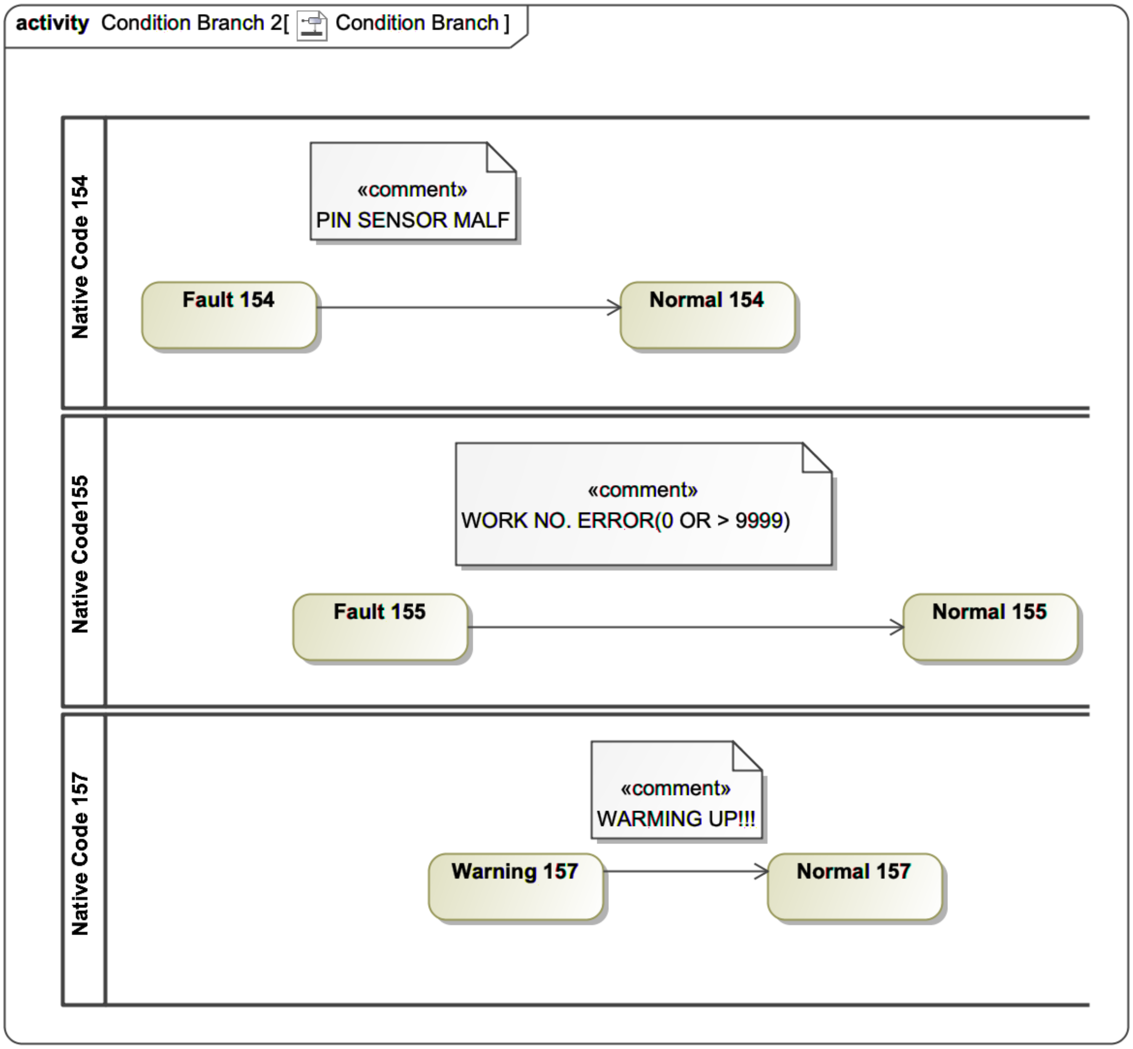

Figure 30: Parallel Conditions

For this example, 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` as given in listing 7, the *DataItem* will be represented in the OPC UA model as a instance of the `MTConditionType` with the attributes presented as properties.

The initial state of the system is given in Table 13 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 11 and the `Normal` initial condition state with no `nativeCode` indicating there are no active conditions.

Listing 11: 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 reports an **Event** MTConditionEventType as shown in Listing 12. A Fault indicates a situation where the piece of equipment is no longer able to continue functioning and needs manual intervention.

Listing 12: 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 13 where a second PLC alarm is active. The native code is different than the previous condition, so a second MTConditionEventType must be reported with a unique **ConditionId** indicated using the **NodeId** in the **Event**.

Listing 13: Path Logic Program Second Fault PLC-155

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

The warning in Listing 14 indicates the machine is warming up and other operations are disabled. This condition has another nativeCode and therefore, like the previous condition, another **Event** must be reported. 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 13.

Listing 14: Path Logic Program Warning PLC-157

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

In Listing 15, 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, 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 13. An `MTConditionEventType` **`Event`** will be reported with its `ActiveState` set to **`False`** and **`Retain`** property set to **`False`**.

Listing 15: Path Logic Program Clear Fault of PLC-154

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

In Listing 16, when the machine finishes warming up, the first condition will be returned to an inactive state. It is indicated by `Normal` and a native code of `PLC-157` and will be handled like the previous case. 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 13 clears the active state of this **`Event`** on Row 6.

Listing 16: Path Logic Program Clear Warning PLC-157

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

Listing 17 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 13 shows the clearing of the final activation and then we clear everything in Row 8.

Listing 17: Path Logic Program Back to Normal, All Clear

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

9 MTConnect OPC UA Types

Clause 9.4: Replace 9.4 with the following:

9.4 Conditions

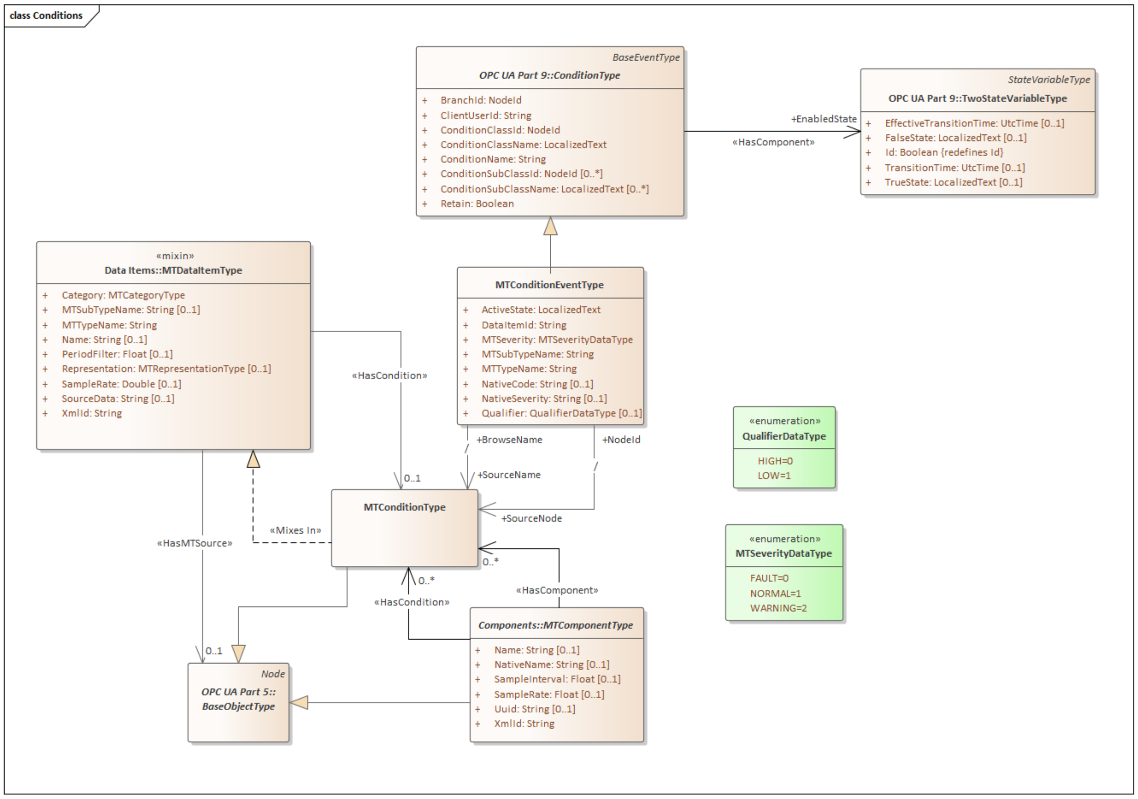
class Conditions
BaseEventType
OPC UA Part 9::ConditionType
+ BranchId: NodeId
+ ClientUserId: String
+ ConditionClassId: NodeId
+ ConditionClassName: LocalizedText
+ ConditionName: String
+ ConditionSubClassId: NodeId [0..*]
+ ConditionSubClassName: LocalizedText [0..*]
+ Retain: Boolean
+EnabledState
«HasComponent»
StateVariableType
OPC UA Part 9::TwoStateVariableType
+ EffectiveTransitionTime: UtcTime [0..1]
+ FalseState: LocalizedText [0..1]
+ Id: Boolean {redefines Id}
+ TransitionTime: UtcTime [0..1]
+ TrueState: LocalizedText [0..1]
«mixin»
Data Items::MTDataItemType
+ Category: MTCategoryType
+ MTSubTypeName: String [0..1]
+ MTTypeName: String
+ Name: String [0..1]
+ PeriodFilter: Float [0..1]
+ Representation: MTRepresentationType [0..1]
+ SampleRate: Double [0..1]
+ SourceData: String [0..1]
+ XmlId: String
MTConditionEventType
+ ActiveState: LocalizedText
+ DataItemId: String
+ MTSeverity: MTSeverityDataType
+ MTSubTypeName: String
+ MTTypeName: String
+ NativeCode: String [0..1]
+ NativeSeverity: String [0..1]
+ Qualifier: QualifierDataType [0..1]
«enumeration»
QualifierDataType
HIGH=0
LOW=1
«enumeration»
MTSeverityDataType
FAULT=0
NORMAL=1
WARNING=2
«HasCondition»
0..1
+BrowseName
+NodeId
+SourceName
+SourceNode
MTConditionType
«Mixes In»
«HasMTSource»
0..*
0..*
«HasCondition»
«HasComponent»
0..1
Node
OPC UA Part 5::
BaseObjectType
Components::MTComponentType
+ Name: String [0..1]
+ NativeName: String [0..1]
+ SampleInterval: Float [0..1]
+ SampleRate: Float [0..1]
+ Uuid: String [0..1]
+ XmlId: String

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.

In MTConnect, the `Condition` provides the meta-data describing the `type`, `subType`, and other properties required by the MTConnect *DataItem*. In response to a *sample request*, the `Conditions` behaves like an event stream. In response to 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 state at a time; the `Condition` is different and can have multiple states and values, as in the case where there are multiple system alarms for a component.

The set of active `Condition` instances are represented in the OPC UA model as **`Events`** and utilize the **Active** and **Retain** attributes to indicate if they are currently of interest. Details are provided in Section 8.4.6.

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 active Condition.

9.4.1 Defintion of MTConditionEventType

The condition type is derived from the UA **ContitionType**. When the Warning or Fault state occurs, an MTConditionEventType **Event** is created with the ActiveState set to **True** and **Retain** set to **True**. The severity is used to represent the MTConnect condition states of Warning and Fault with the values of 500 and 1000 respectively.

A new **NodeId** will be created for every unique instance of the MTConnect Condition reported. When the Condition goes back to Normal, the ActiveState is set to **False** and **Retain** is also set to **False** with the **NodeId** of the associated Condition. If multiple MTConnect Conditions have been cleared at the same time, all currently active MTConditionEventType **Event**s will need to deactivated.

The MTConditionEventType must set the **BaseEvent SourceNode** to the related MTConditionType that represents the meta-data for this Condition.

The MTConditionEventType will never be instantiated in the ***AddressSpace*** as an **Object**.

Table 78: MTConditionEventType Definition

Attribute	Value				
BrowseName	MTConditionEventType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModelingRule**
Subtype of ConditionType (See [UA Part 09] Documentation)					
HasProperty	Variable	ActiveState	LocalizedText	PropertyType	Mandatory
HasProperty	Variable	DataItemId	String	PropertyType	Mandatory
HasProperty	Variable	MTSeverity	MTSeverityDataType	PropertyType	Mandatory
HasProperty	Variable	MTSubTypeName	String	PropertyType	Mandatory
HasProperty	Variable	MTTypeName	String	PropertyType	Mandatory
HasProperty	Variable	NativeCode	String	PropertyType	Optional
HasProperty	Variable	NativeSeverity	String	PropertyType	Optional
HasProperty	Variable	Qualifier	QualifierDataType	PropertyType	Optional

9.4.1.1 Referenced Properties and Objects

- DataItemId : String: The identifier attribute of the dataitem that represents the originally measured value of the data referenced by this data item.

- **Allowable Values** for MTSeverityDataType

Table 79: MTSeverityDataType Enumeration

Name	Index	Description
FAULT	0	Fault value for a condition element.
NORMAL	1	Normal value for a condition element.
WARNING	2	Warning value for a condition element.

- 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. The native code (usually an alpha-numeric value) generated by the controller of a piece of equipment or the element.

- 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. If the piece of equipment designates a severity level to a fault, nativeseverity reports that severity information to a client software application.

- Qualifier : QualifierDataType: qualifier provides additional information regarding a fault state associated with the measured value of a process variable.

- **Allowable Values** for QualifierDataType

Table 80: QualifierDataType Enumeration

Name	Index	Description
HIGH	0	High qualifier value for a condition element.
LOW	1	Low qualifier value for a condition element.

- SourceName : MTConditionType: The **SourceName** is mapped to the **BrowseName** of the MTConditionType.
- SourceNode : MTConditionType: The **SourceNode** is mapped to the **NodeId** of the MTConditionType.

9.4.2 Defintion of **MTConditionType**

An MTConditionType instance will be created for event MTConnect *DataItem* with a category of CONDITION.

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.

The information and data reported from a piece of equipment for those DataItems defined with a category of Condition.

9.4.2.1 Dependencies and Relationships

- Mixes in MTDataItemType, see See section **??**

Table 81: `MTConditionType` Definition

Attribute	**Value**				
BrowseName	MTConditionType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**Type-Definition**	**Modeling-Rule**
Subtype of BaseObjectType (See [UA Part 05] Documentation)					
HasProperty	Variable	Category	MTCategoryType	PropertyType	Mandatory
HasProperty	Variable	MTSubTypeName	String	PropertyType	Optional
HasProperty	Variable	MTTypeName	String	PropertyType	Mandatory
HasProperty	Variable	Name	String	PropertyType	Optional
HasProperty	Variable	PeriodFilter	Float	PropertyType	Optional
HasProperty	Variable	Representation	MTRepresentationType	PropertyType	Optional
HasProperty	Variable	SampleRate	Double	PropertyType	Optional
HasProperty	Variable	SourceData	String	PropertyType	Optional
HasProperty	Variable	XmlId	String	PropertyType	Mandatory
HasMTSource	Object	<BaseObject>	BaseObjectType		Optional
HasMTComposition	Object	<MTComposition>	MTCompositionType		Optional
HasMTSubClassType	Object	<MTDataItemSubClass>	MTDataItemSubClassType		Optional
HasCondition	Object	<MTCondition>	MTConditionType		Optional
HasComponent	Object	Constraints	MTConstraintType		Optional
HasMTClassType	Object	<MTDataItemClass>	MTDataItemClassType		Mandatory

Annex A MTConnect Namespace and Mappings (normative)

A.1 Namespace and identifiers for MTConnect Information Model

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

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

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

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

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

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

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

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

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

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

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

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

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